Engenharia Informática e Multimédia

Infraestruturas Computacionais Distribuídas

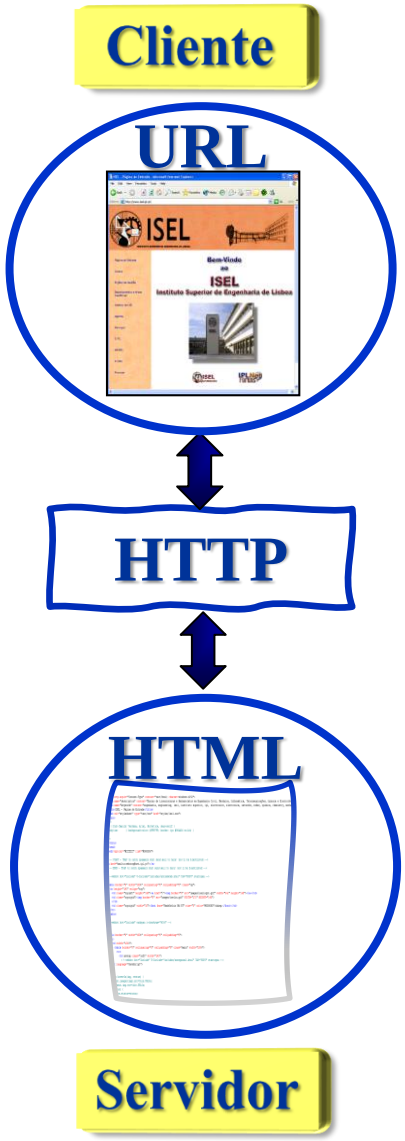
World Wide Web

Departamento de Engenharia Electrónica e Telecomunicações e de Computadores

Versão 1.3

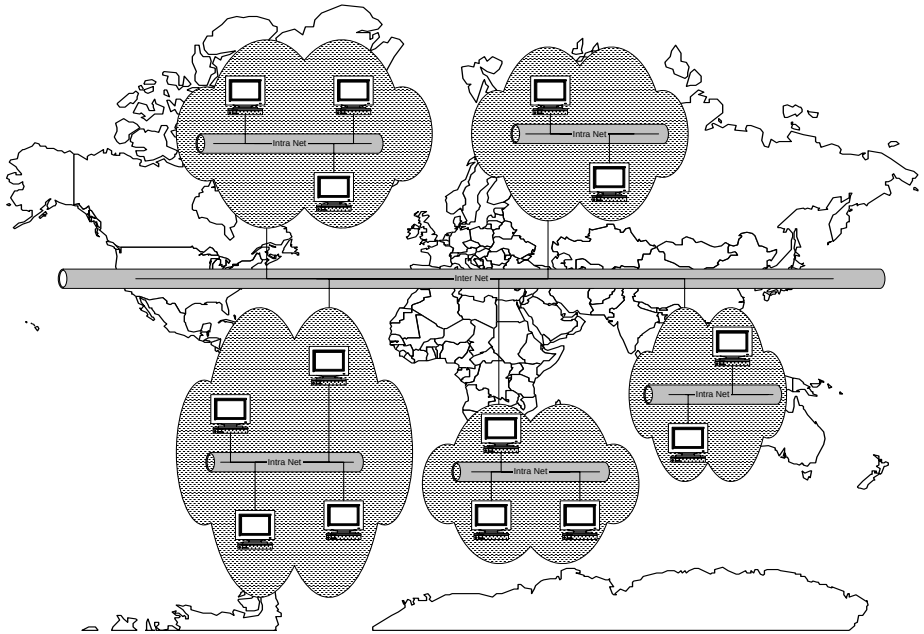
Agenda

- Introdução
  - *Internet*, *Intranet* e *Extranet*
  - Noção de Serviço
- *World Wide Web*, WWW
  - Localizador, URL
  - Protocolo, HTTP
  - Linguagem, HTML

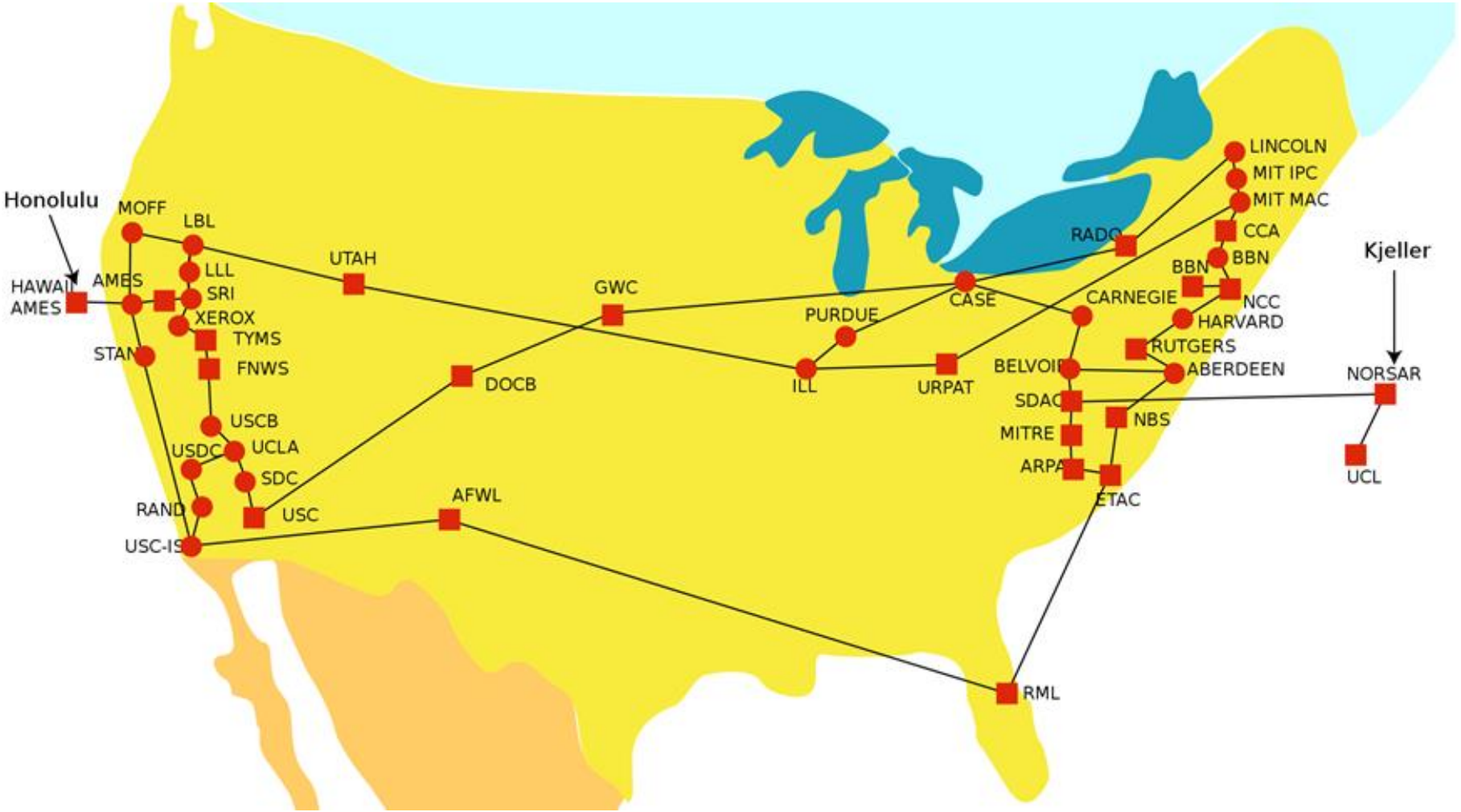


Introdução: *Internet*

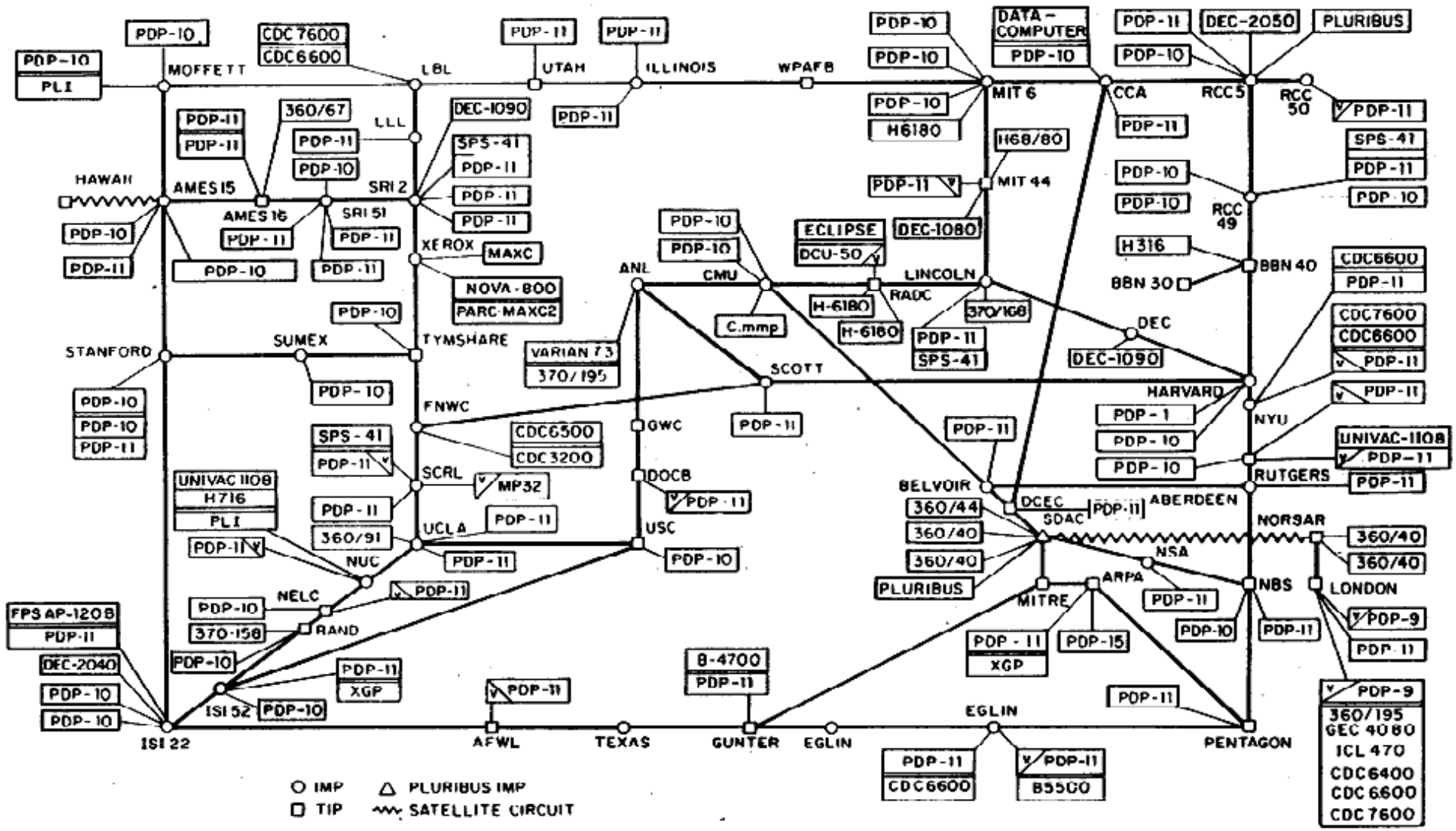
- *Local Area Network* (LAN): é uma rede local onde os computadores estão ligados e podem partilhar o mesmo acesso á *Internet*
- *Internet*: é um aglomerado de redes, interligadas e globalmente distribuídas
- Origem: departamento de defesa dos Estados Unidos com a criação da ARPANET em 1969
- Motivação: garantir comunicações táticas na eventualidade de falha, em situação de catástrofe, da rede telefónica



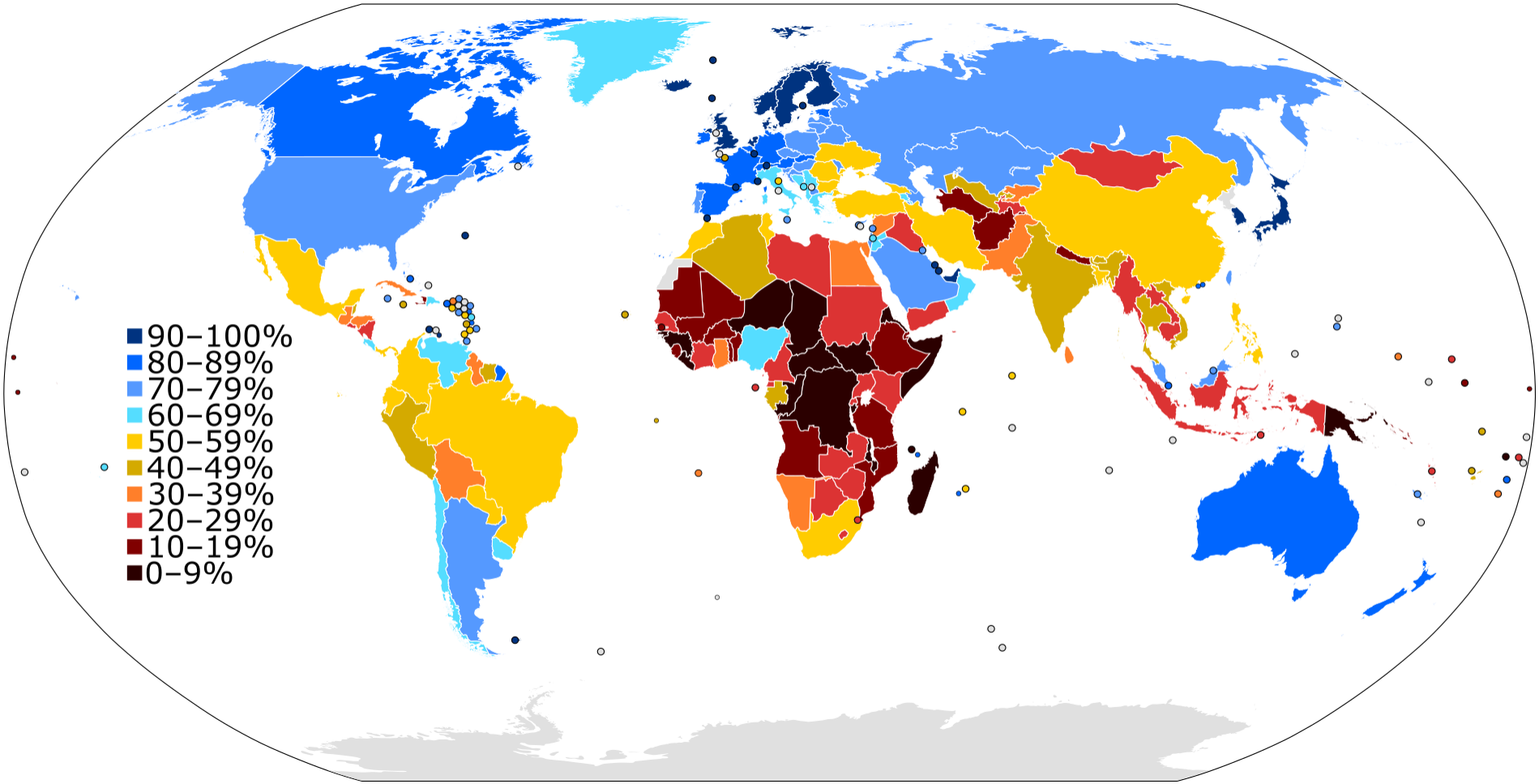
Introdução: *ARPANET* (1974)



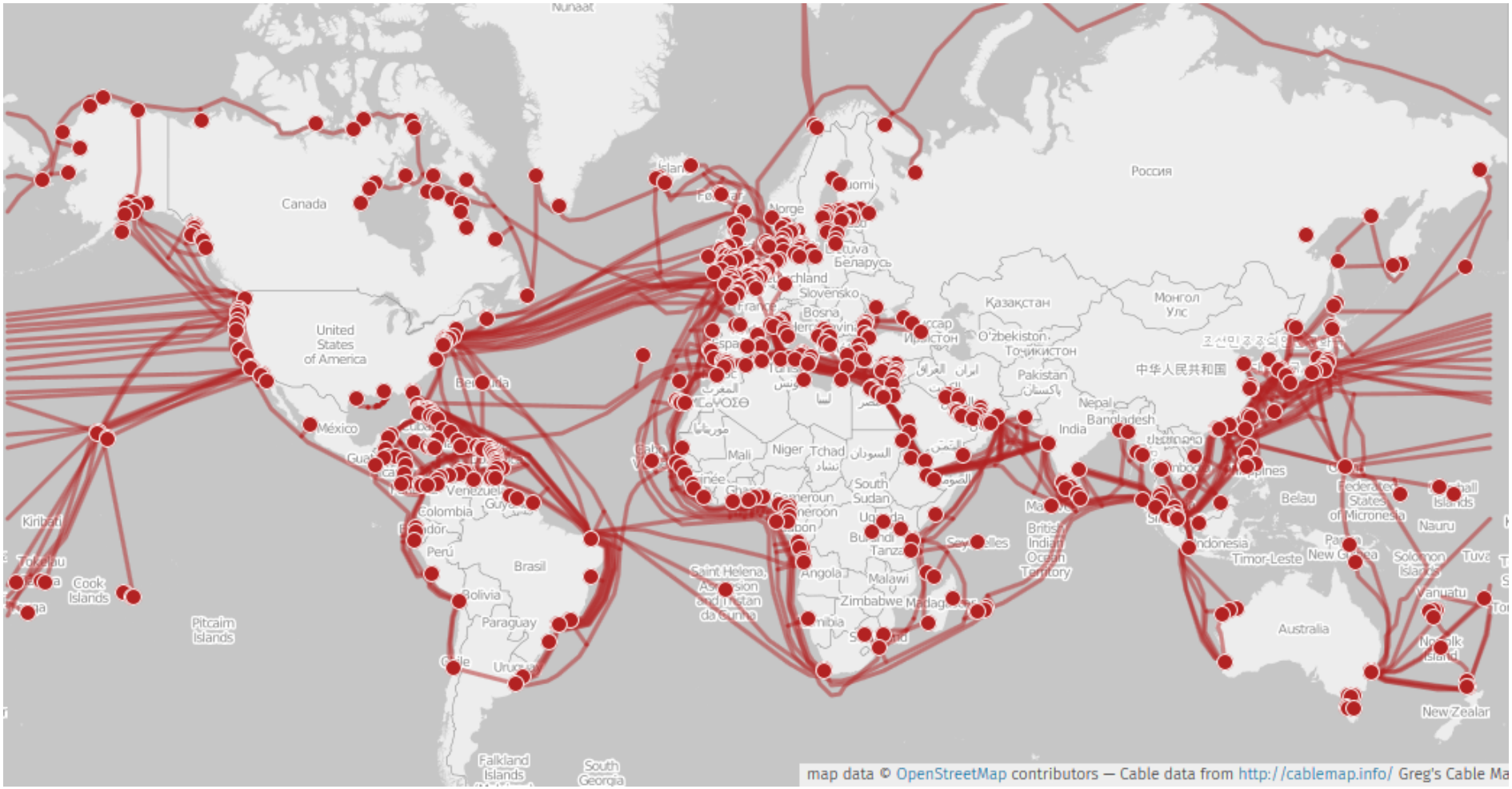
Introdução: *ARPANET* (1977)



Introdução: Utilização da *Internet*



Introdução: Rede de Cabos Submarinos



Introdução: *Intranet/Extranet*

- *Intranet*: é uma rede de computadores privada (interna) baseada em protocolos da *Internet*

- *Extranet*: é uma extensão da Intranet que dá privilégios de acesso a utilizadores externos: clientes ou fornecedores.

  - Exige ligações fiáveis e comunicações protegidas
  - Permite organizar negócios interligando empresas parceiras
  - Facilita o acesso à parte privada de um portais institucionais
  - Os protocolos abertos da Internet, dispensam uniformizar o sistema operativo, *hardware* ou *browser*.

Introdução: Noção de Serviço

Nome	*Porto/Protocolo*	*Descrição*
FTP	20,21 / TCP	Transferência de ficheiros
TELNET	23 / TCP	*Shell* Remota
SSH	22	*Secure Shell*, Transferência de ficheiros, *Port forward*
SMTP	25 / TCP	Envio de email
POP3	110 / TCP	Consulta de email
RPC	111 / TCP|UDP	*Remote Procedure Call*
HTTP	80 / TCP	Servidor WEB
HTTPS	443 / TCP	Servidor WEB seguro
NFS	2049 / TCP|UDP	Partilha de ficheiros em rede
IRC	6667 / TCP	*Internet Relay Chat Protocol*

https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

Internet vs. *World Wide Web*

A *Internet* é uma Rede de Redes

"The Internet is a global system of interconnected computer networks that interchange data by packet switching using the standardized Internet Protocol Suite (TCP/IP)"

A *Web* é um Espaço de Informação

"The World Wide Web (WWW, or simply Web) is an information space in which the items of interest, referred to as resources, are identified by global identifiers called Uniform Resource Identifiers (URI)"

Web: Introdução

- A *Wordl Wide Web* (WWW ou Web) é uma aplicação distribuída de larga escala suportada por serviços (ex. HTTP) da *Internet*
- Teve origem no CERN no final da década de 80, devido à necessidade dos investigadores partilharem os resultados das suas experiências
- A *Web* foi inventada Tim Berners-Lee, um investigador do CERN, que desenvolveu a linguagem HTML (*HiperText Markup Language*) e as infraestrutura para a utilizar
- A HTML facilita a representação de documentos geridos por servidores *Web* que podem ser acedidos por (navegador/*browser)* clientes *Web* usando o conceito de *hyperlink*

Web: Introdução

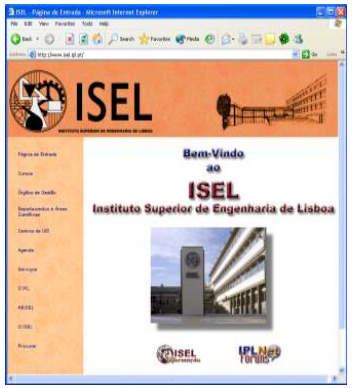
- A *Web* facilita a partilha transparente de recursos, ultrapassando dificuldades associadas à heterogeneidade tecnológica, recorrendo a:
  - Esquema para localizar os recursos (ex. URL)
  - Protocolo para aceder aos recursos (ex. HTTP)
  - Linguagem para estruturar os recursos (ex. HTML)

Web: Interação

- Interação baseada na arquitetura Cliente – Servidor
- A comunicação, ao nível da camada de transporte, é baseada em TCP

  Transmission Control Protocol
- A comunicação, ao nível da camada aplicacional, é baseada no HTTP

  HyperText Transfer Protocol
  - Pedido/Envio de mensagens de texto para/pelo servidor



Refere Recurso

Pedido HTTP

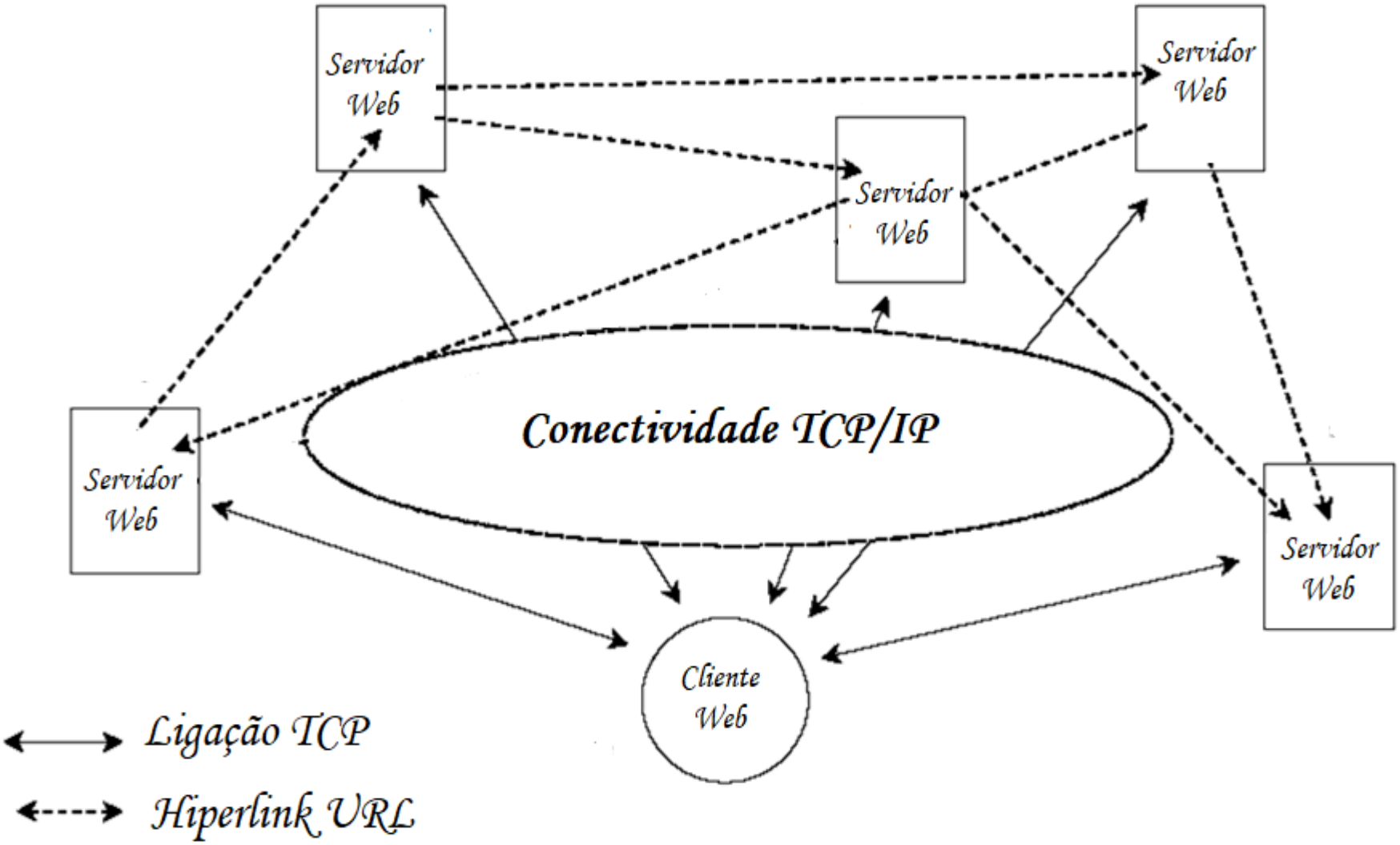
Contém Recurso

Resposta HTTP

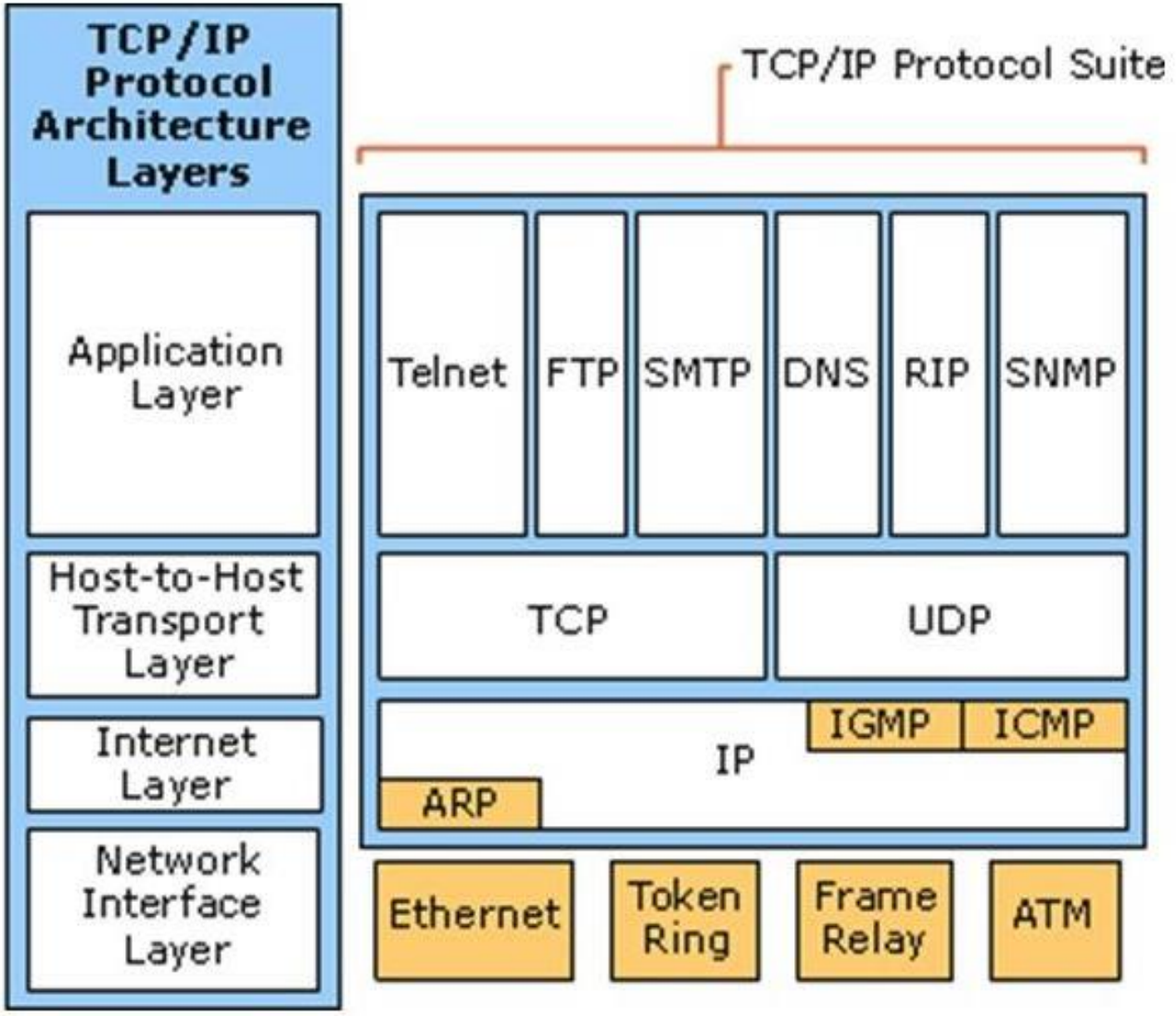
Cliente *Web*

Servidor Web

Web: Cliente/Servidor

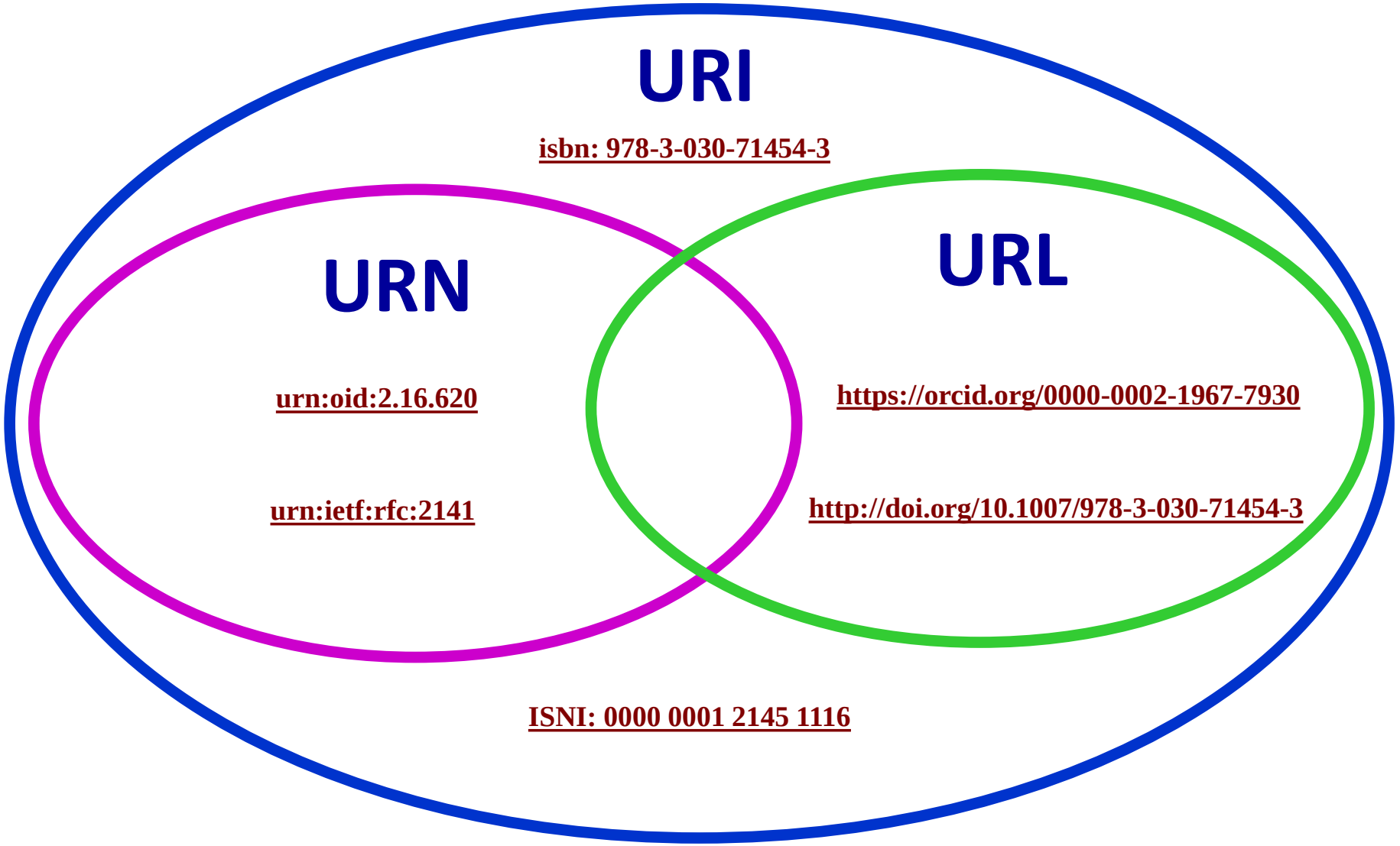


Família de Protocolos TCP/IP

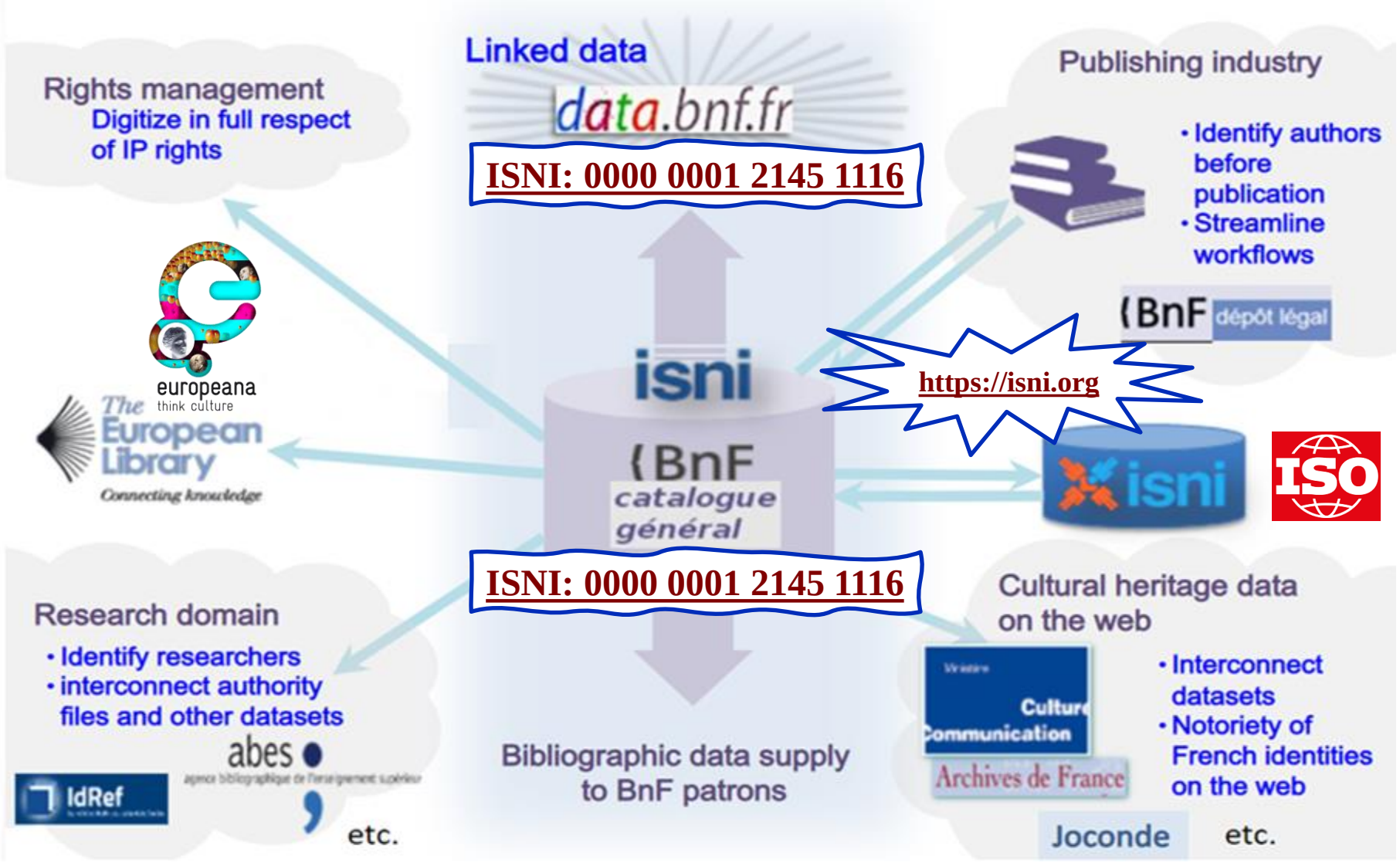


Identificadores de Recursos

URI: Identificador Uniforme de Recurso



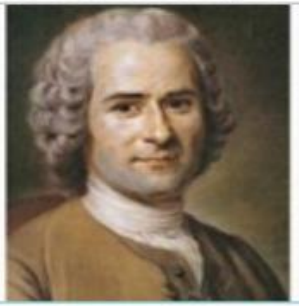
Exemplo: Biblioteca Nacional de França



ISNI: *International Standard Name Identifier*

ISNI: 0000 0001 2145 1116

Jean-Jacques **Rousseau** (1712-1778)



WIKIPEDIA

http://fr.wikipedia.org/wiki/Jean-Jacques_Rousseau

https://www.wikidata.org/wiki/Q6527

WIKIDATA

VIAF BAV BNC BNE BNF DNB EGAXA ICCU JPG LAC LC LNB NDL NKC NLI NLP NSK NTA NUKAT PTBNP RERO SELIBR SUDOC SWNL WKP BOWKER MUBZ TEL ZETO

isni

Other French cultural datasets

IdRef

www.idref.fr/028202368

Universities' Union Catalog

BnF catalogue général

https://catalogue.bnf.fr/ark:/12148/cb119228797

Authority file

data.bnf.fr

https://data.bnf.fr/en/11922879/jean-jacques_rousseau/

LOD service

ISNI: Exemplos Relacionados

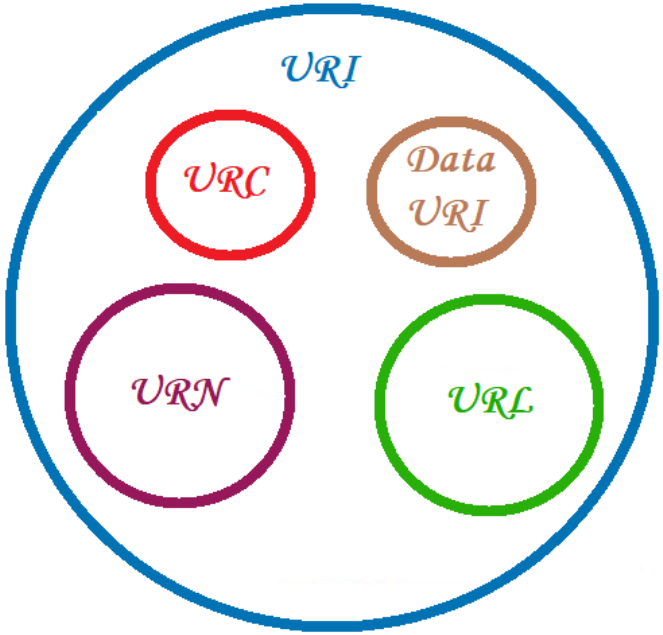
- *Digital Object Identifier* (DOI)
  - https://www.doi.org
  - https://www.doi.org/demos.html
  - https://doi.org/10.1109/5.771073
- *International Standard Audiovisual Number* (ISAN)
  - https://www.isan.org
- *International Standard Book Number* (ISBN)
  - http://isbn.org
- *International Standard Recording Code (*ISRC)
  - https://www.isrc.com
- *International Standard Serial Number (*ISSN)
  - https://www.issn.org
- *Standard Musical Work Code* (ISWC)
  - https://www.iswc.org
- *Open Researcher and Contributor Identifier* (ORCID)
  - https://orcid.org

Timothy J Berners-Lee

~~Porfírio~~ Filipe

URI: Identificador Uniforme de Recurso

- *Uniform Resource Identifier* (URI)
  - Identificação/localização de recursos (RFC 1630, 3986)



- Sintaxe genérica

  <Parte Genérica> : <Parte Específica>

  isbn:0-486-27557-4

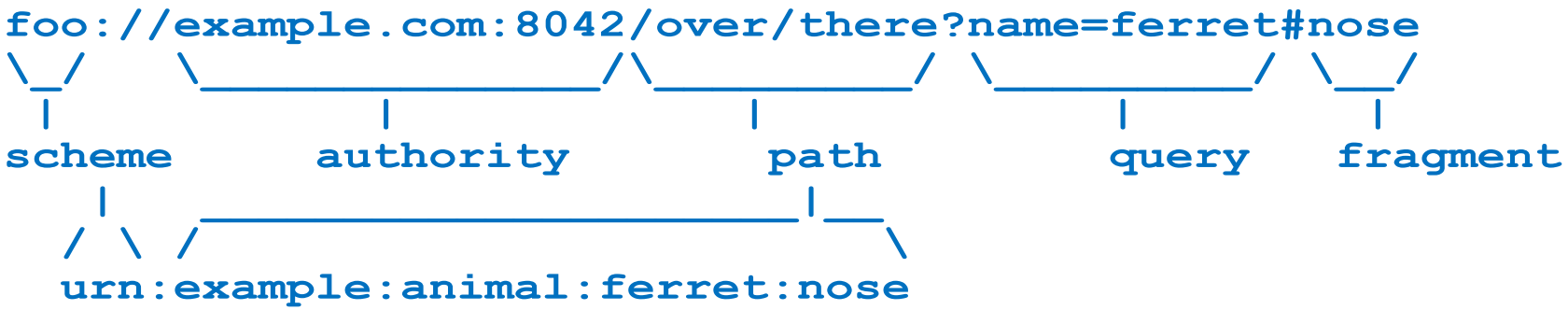
  view-source:http://isel.pt/hello.html

  data:,Hello%2C%20World

- Tipos de URI (RFC 3305):
  - *Uniform Resource Name* (URN)
    - Identificação persistente, independente da localização (RFC 2141)
  - *Uniform Resource Locator* (URL)
    - Identificação por localização (RFC 1738)

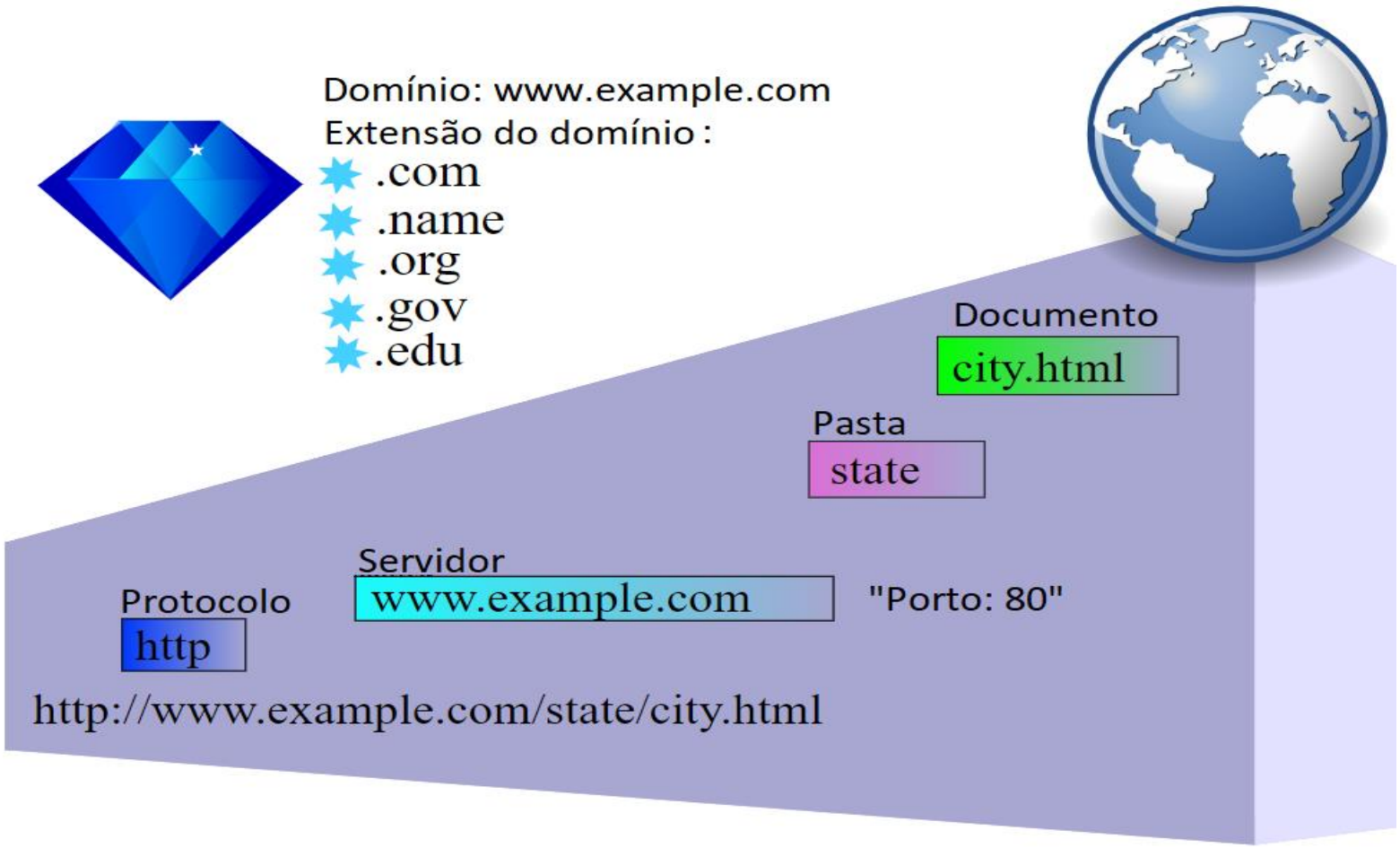
URI: Exemplos

- ftp://ftp.is.co.za/rfc/rfc1808.txt
- ldap://[2001:db8::7]/c=GB?objectClass?one
- mailto:John.Doe@example.com
- news:comp.infosystems.www.servers.unix
- tel:+1-816-555-1212
- telnet://192.0.2.16:80/
- urn:oasis:names:specification:docbook:dtd:xml:4.1.2
- data:text/html,<script>alert('Hello');</script>



Uniform Resource Identifier (URI): Generic Syntax

URL: Localizador Uniforme de Recursos

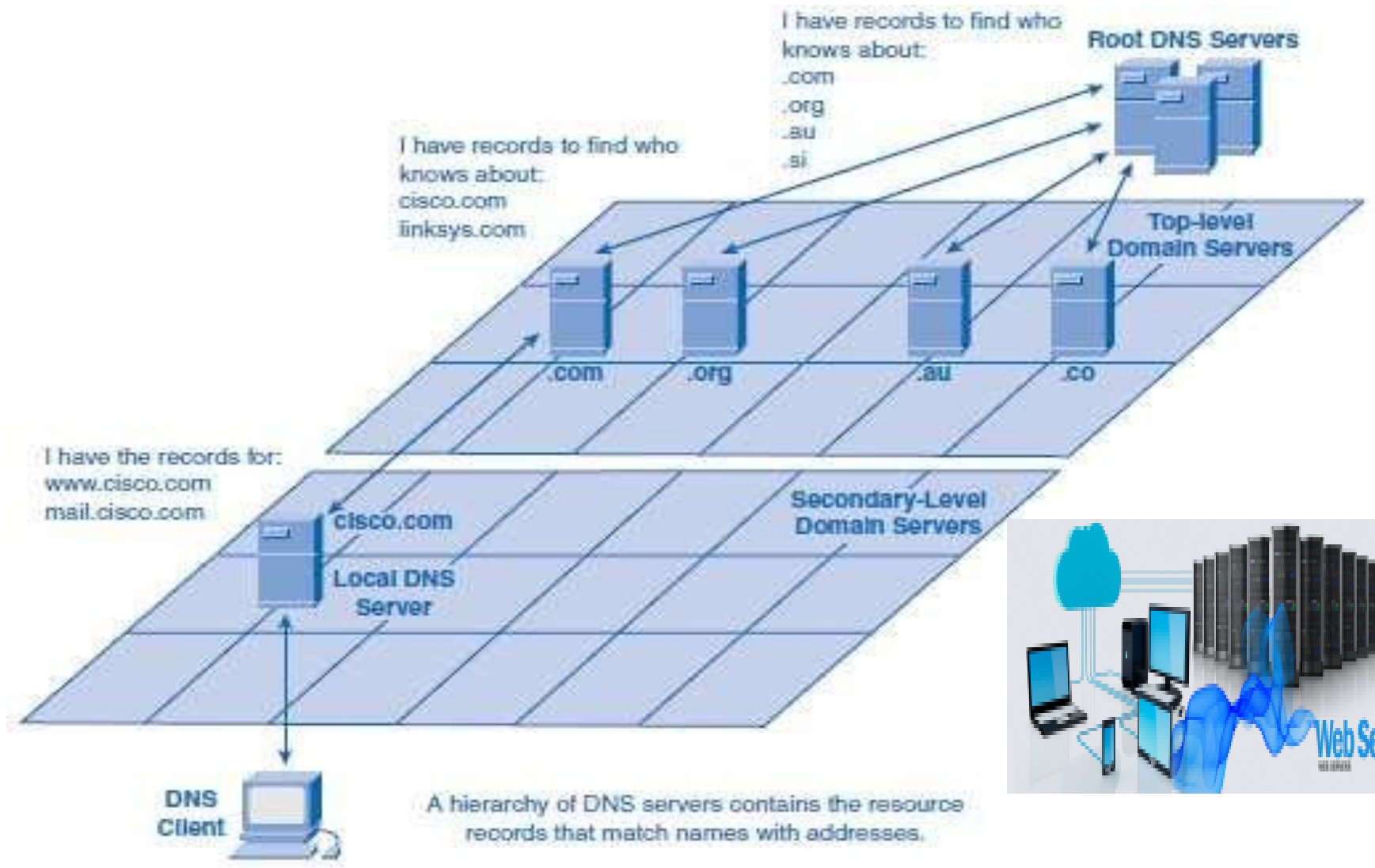


Carateres Reservados

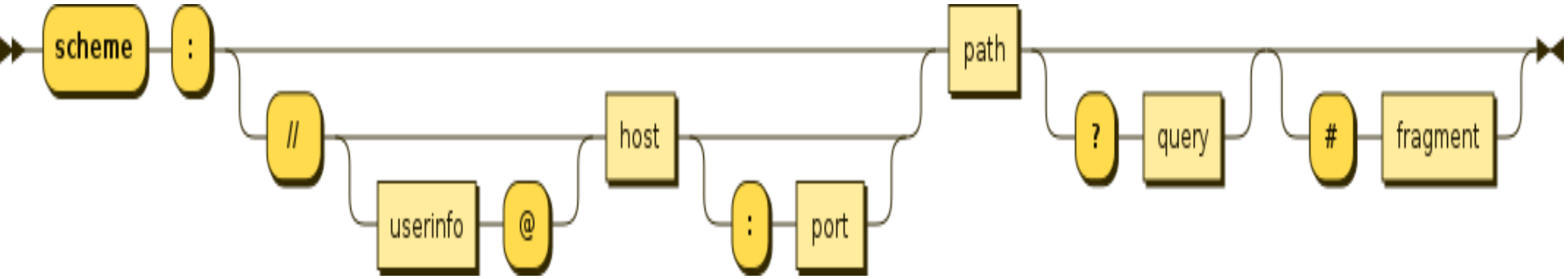
Espaço = %20

! * ' ( ) ; : @ & = + $, / ? # [ ]

URL: Domínios, Resolução de Nomes



URL: Exemplos



- http://www.isel.pt:8080/resource?foo=bar
- http://username:password@example.com/city.html#mayor
- http://datatracker.ietf.org/doc/html/rfc2616#section-3.2.2
- http://www.ietf.org/rfc/rfc5234.txt
- ftp://username:password@example.com/download.zip
- mailto:doe@isel.pt?subject=ol%E1&body=boa%20tarde
- file://home/user/file.txt
- file:///C|\foo\bar ou file:///C:/foo/bar
- https://www.example.com:8080/main-service
- https://www.w3.org
- /other/link.html (um URL relativo)

W3C: URI Testing

URL: Mais Exemplos

File Transfer Protocol (FTP)

ftp://ftp.ncnu.edu.tw/

ftp://ftp.ncnu.edu.tw/JavaDownload/Docs/

ftp://anonymous:guest@ftp.ncnu.edu.tw/

ftp://ycchen@www.im.ncnu.edu.tw/

Gopher Protocol (Gopher)

gopher://gopher.nsysu.edu.tw/11/traveler/train

Electronic Mail (Mailto)

mailto:username@ncnu.edu.tw

mailto:username@ncnu.edu.tw?subject=Hello!

Usenet News (News)

news:tw.bbs.rec.travel

news:*

Telnet to Remote Host (Telent)

telnet://bbs.ee.cycu.edu.tw/

telnet://guest@bbs.ncnu.edu.tw/

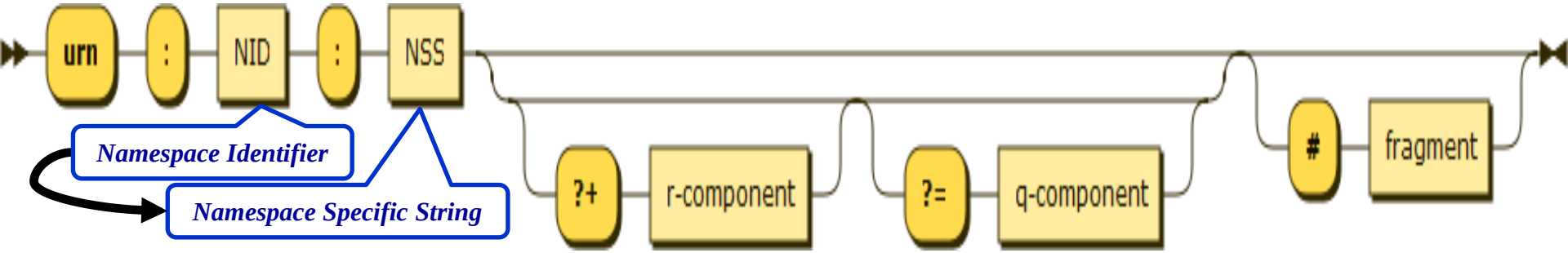
Host-Specific File Names (File)

file:///C|/My Documents/

URL	URI
URL significa Localizador Uniforme de Recurso	URI significa Identificador Uniforme de Recurso
URL é um subconjunto de URI que especifica onde um recurso existe e o mecanismo para o obter	Um URI é um superconjunto de URL que identifica um recurso por URL ou URN (Nome Uniforme de Recurso) ou ambos
O objetivo do URL é identificar a localização ou o endereço de um recurso	O objetivo do URI é identificar um recurso e diferenciá-lo de outros recursos usando o nome ou a localização
URL é usado para localizar recursos na Web	URI é usado em HTML, XML e XSLT (e mais)
NO URL o esquema deve ser um protocolo, tal como: HTTP, FTP, HTTPS, FILE, FTP, etc.	No URI, o esquema pode ser qualquer coisa, tal como: protocolo, especificação, nome, etc.
As informações do protocolo estão no URL	Não existem informações do protocolo no URI
Exemplo de URL: https://isel.pt	Exemplo de URI: urn:isbn:0-486-27557-4
Contém componentes como protocolo, domínio, caminho, *hash*, sequência de consulta, etc.	Contém componentes como esquema, autoridade, caminho, consulta, componente de fragmento, etc.
Todos os URLs podem ser URIs	Nem todos os URIs são URLs, uma vez que um URI pode ser um nome ou um localizador

URN: Nome Uniforme de Recurso

- Identifica um recurso por um nome único e persistente, que é independente da sua localização (RFC 2141)



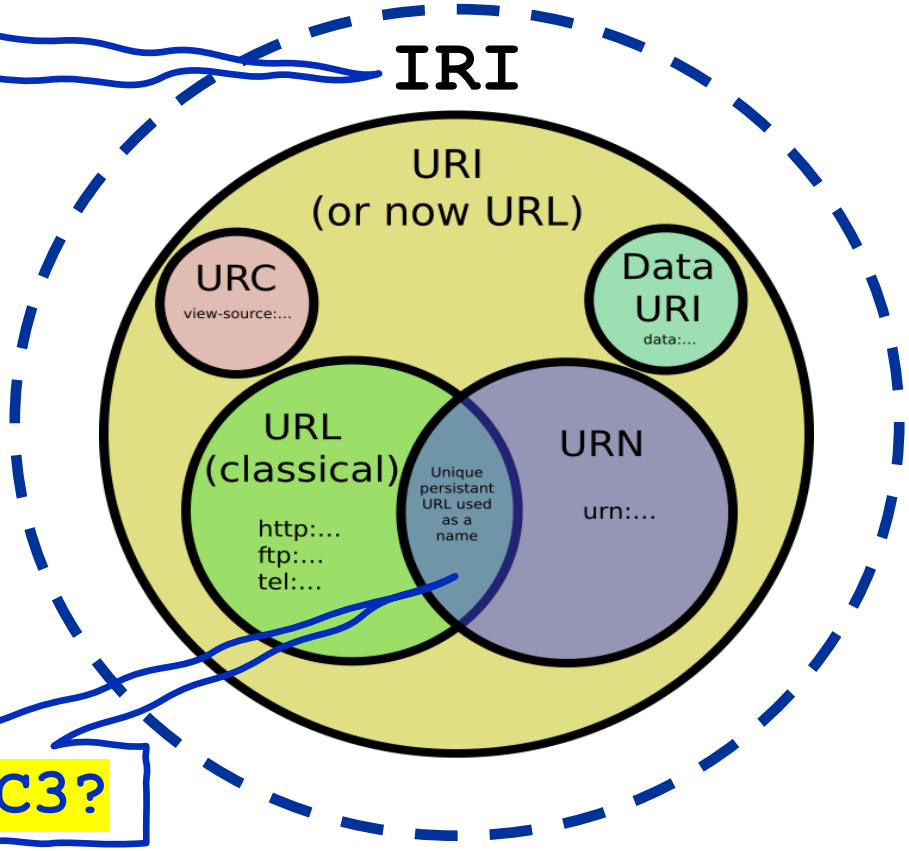
- Identifica de forma global e exclusiva (ID universal)
  - urn:**uuid**:6e8bc430-9c3a-11d9-9669-0800200c9a66
- Identifica um espaço de nomes XML
  - urn:**xmlns**:perfil="www.tutorial.pt/profile"
- Identifica um documento como um tipo de livro
  - urn:**publishing**:book

URI vs. IRI

IRI é um superconjunto de URI (IRI ⊃ URI)
URI é um superconjunto de URL (URI ⊃ URL)
URI é um superconjunto de URN (URI ⊃ URN)
URL e URN são disjuntos (URL ∩ URN = ∅)

RFC 3987
Internationalized
Resource Identifier

Codificação Unicode
Substituição dos URIs

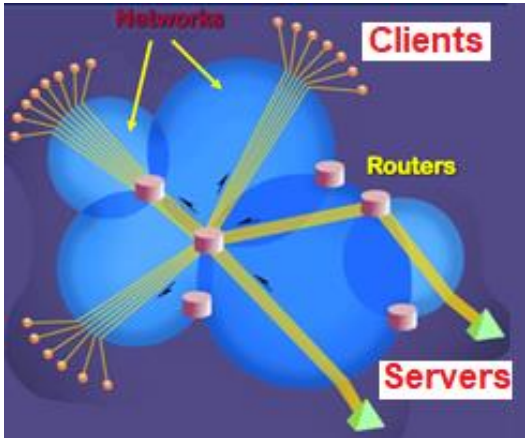


Visão do WC3?

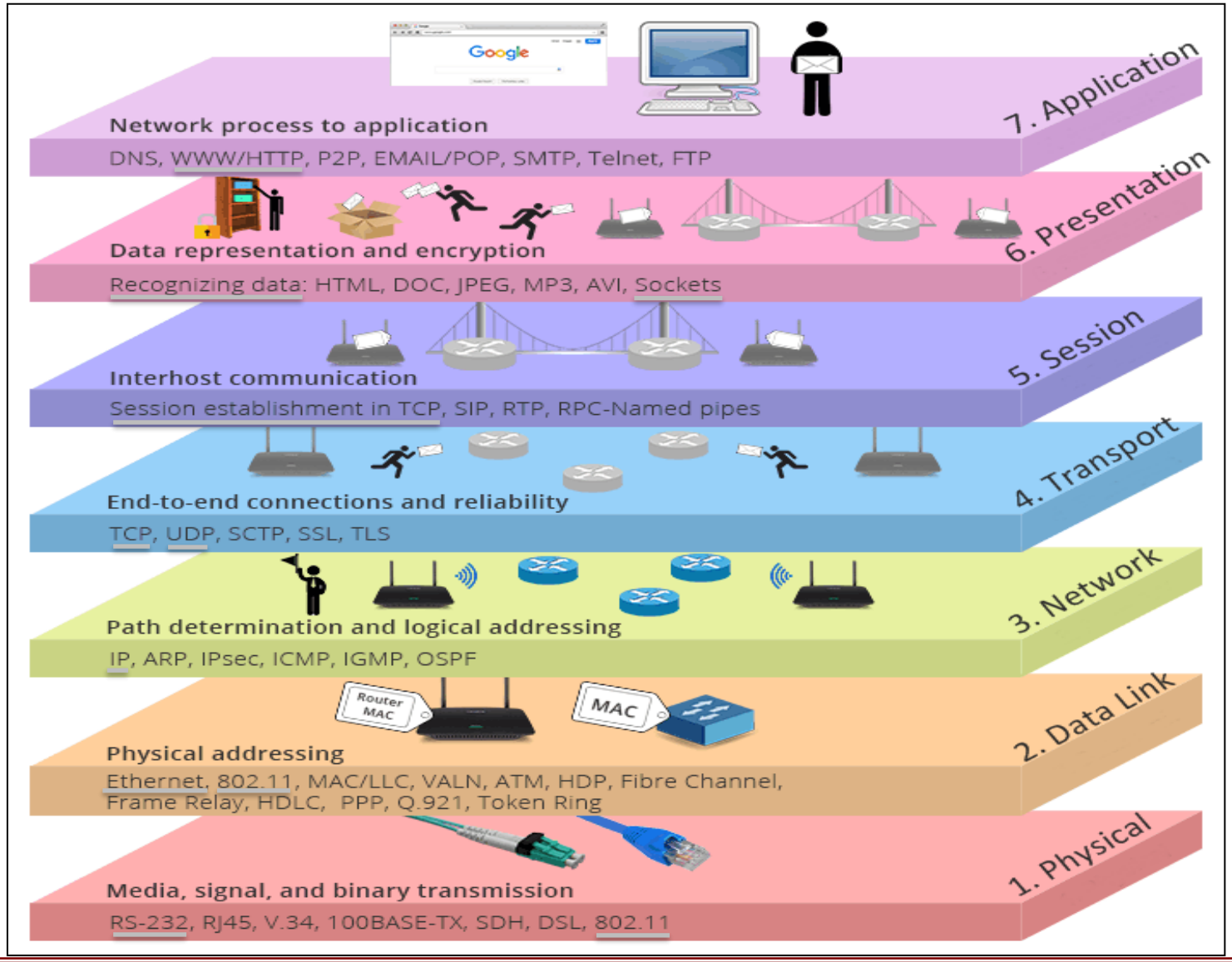
Protocolo HTTP

(HyperText Transfer Protocol)

Arquitetura Geral

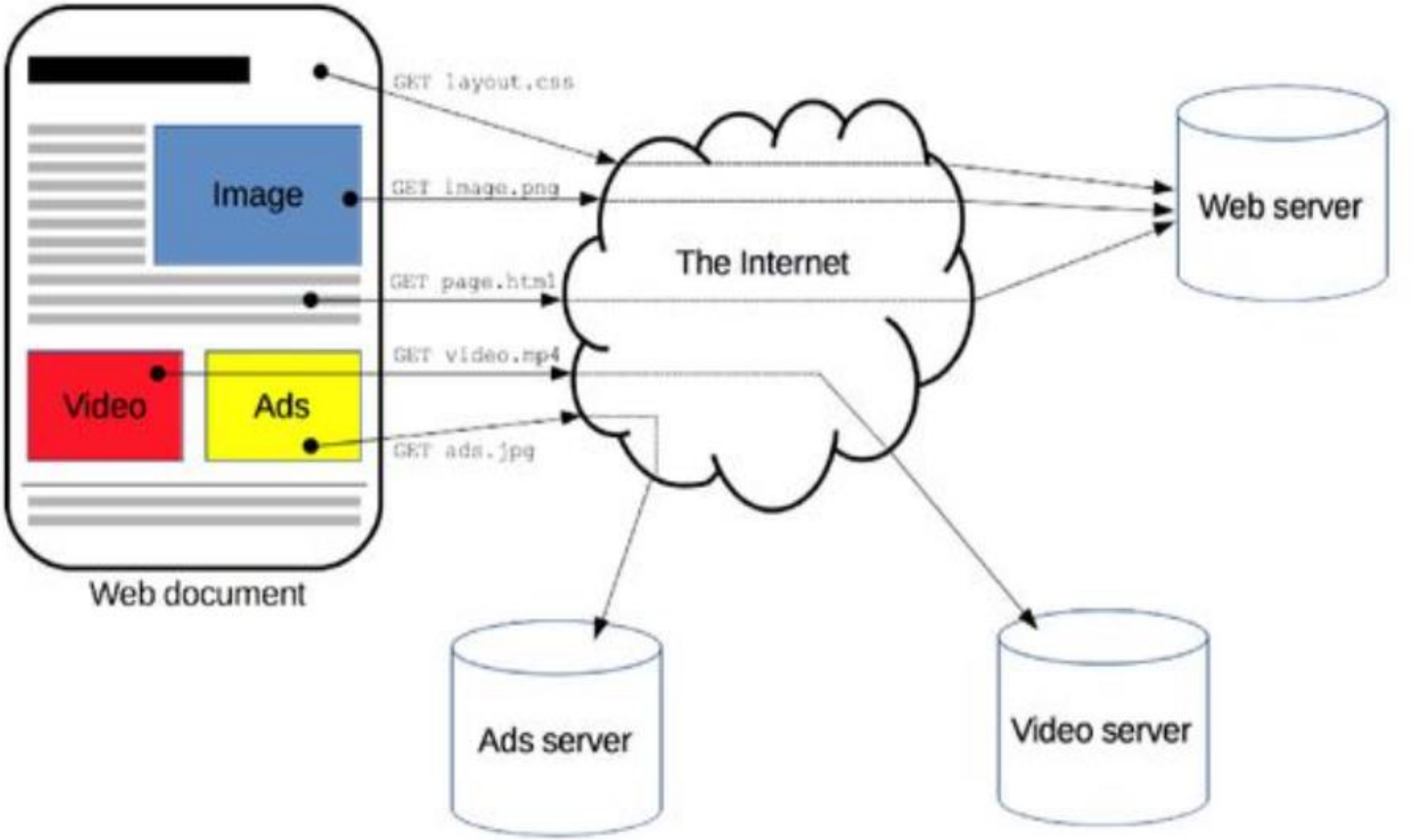


Modelo OSI

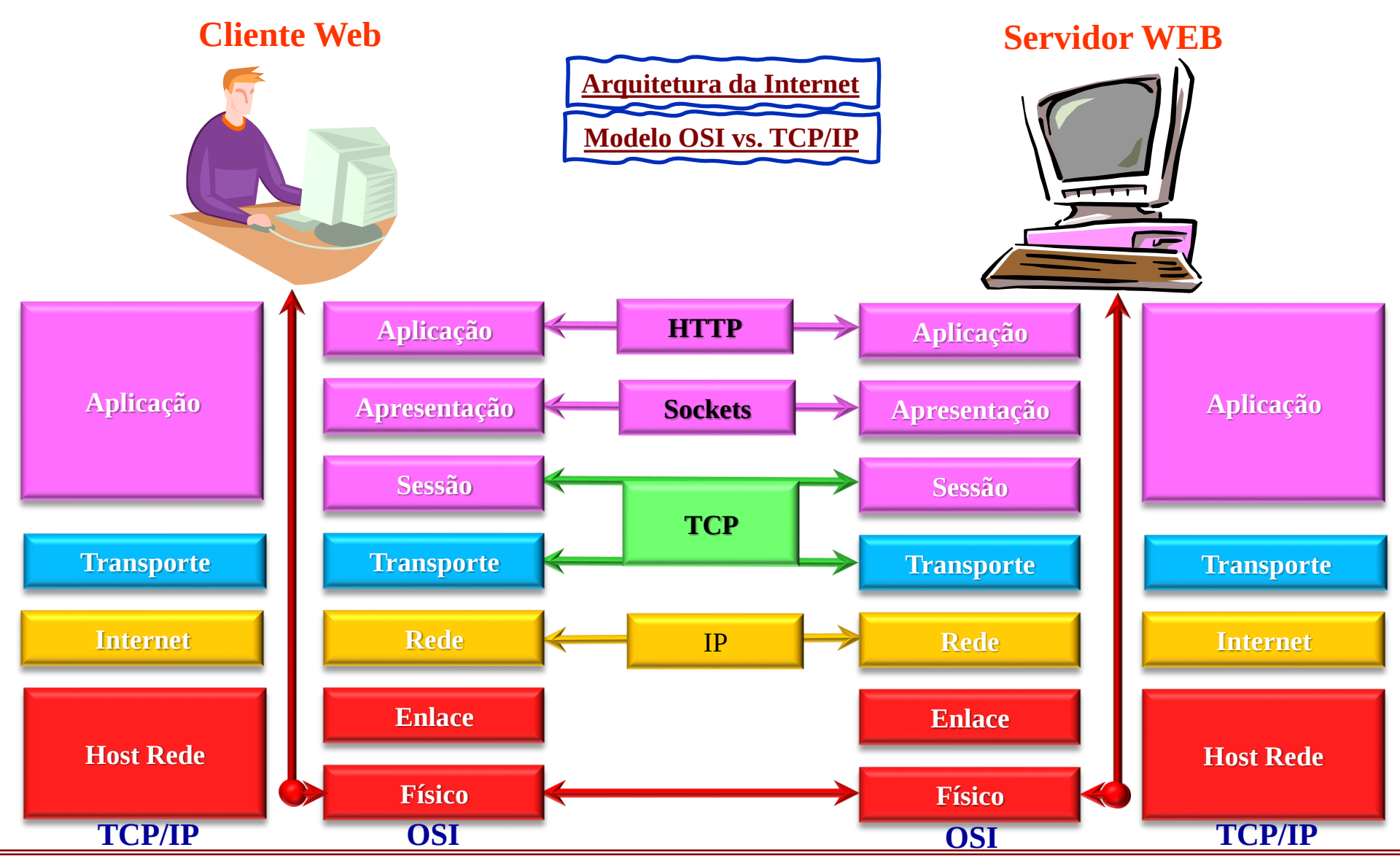


Modelo OSI

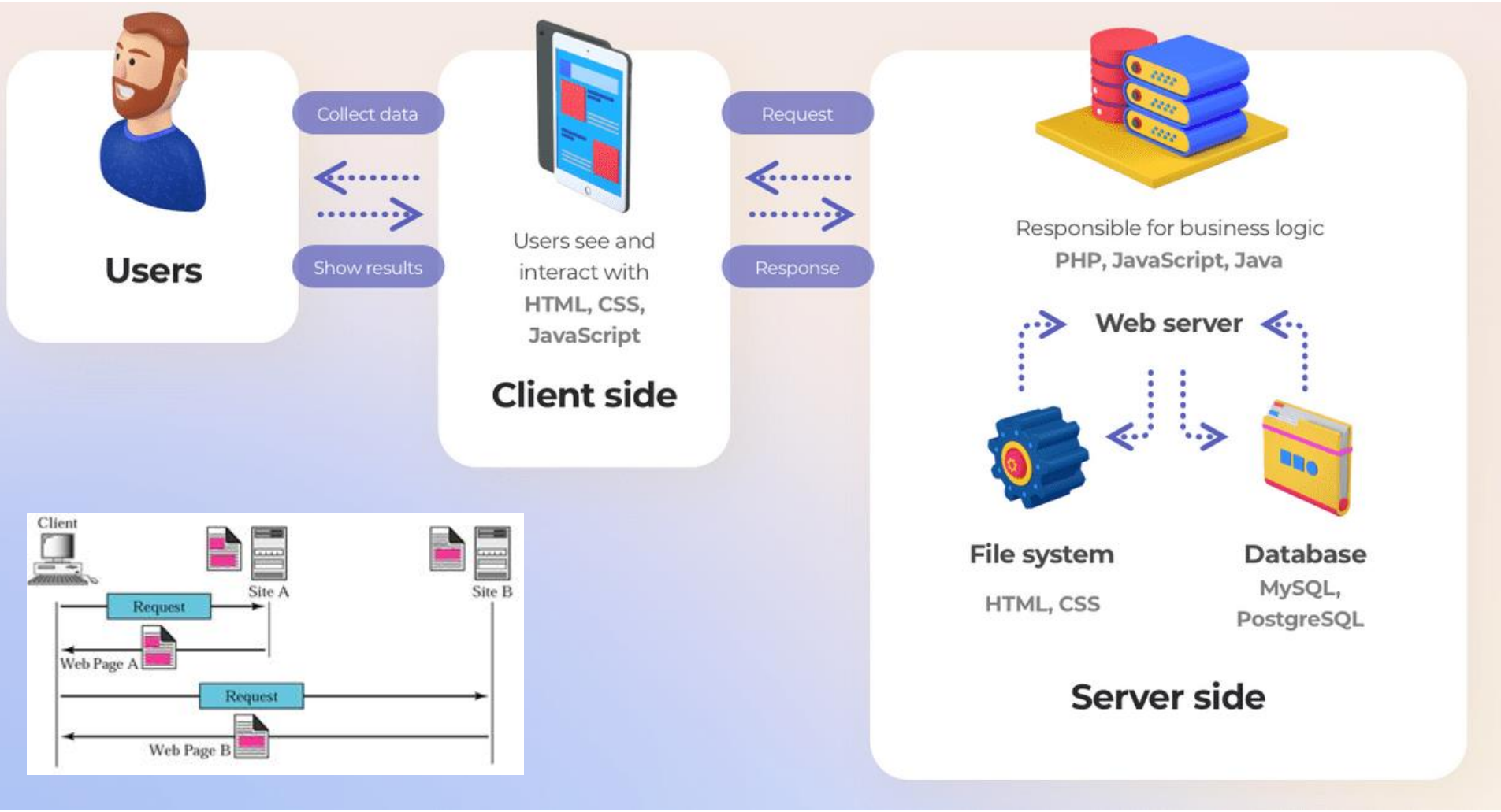
HTTP: Funcionamento Geral



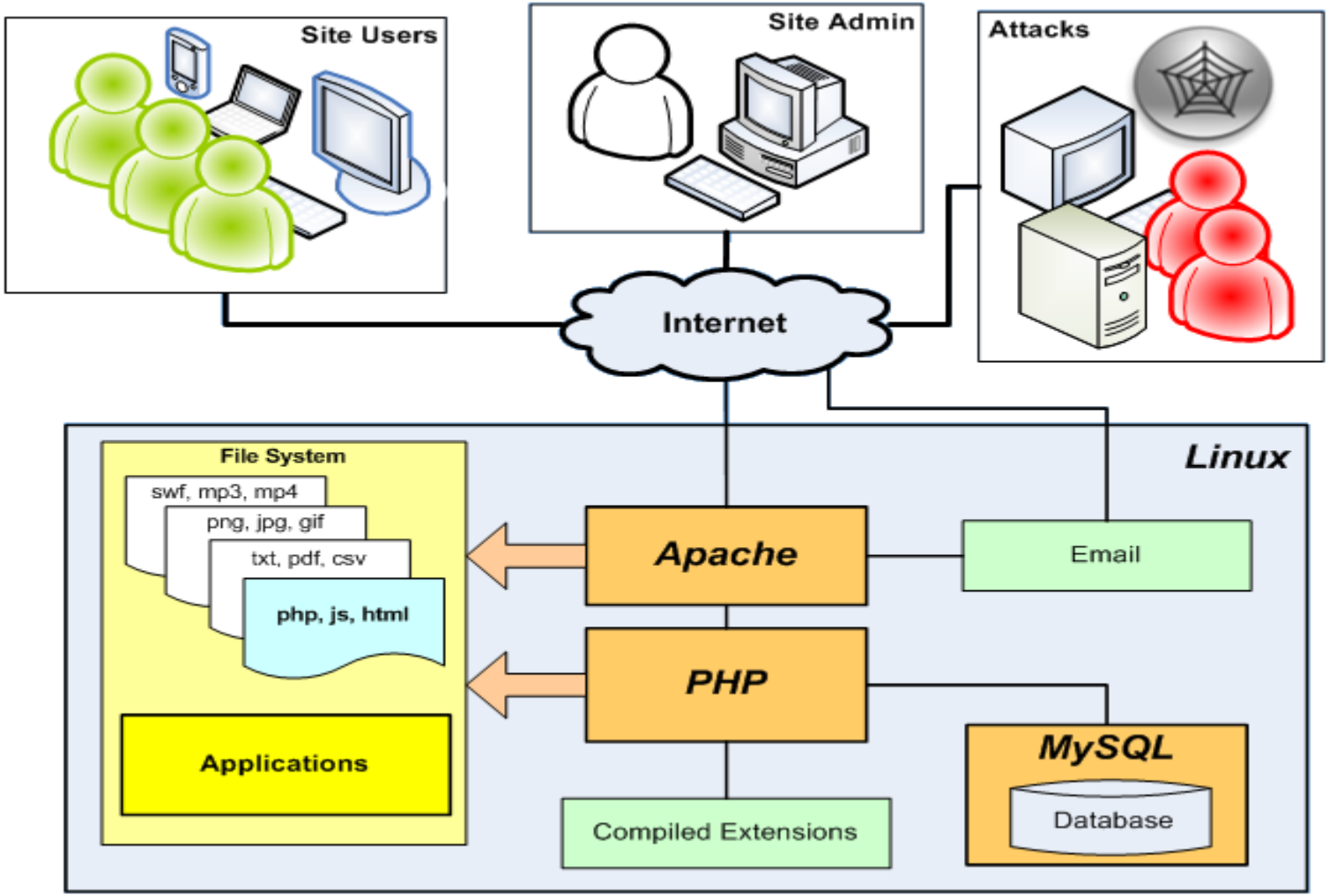
HTTP : Modelo OSI vs. TCP/IP



HTTP: Arquitetura Aplicacional

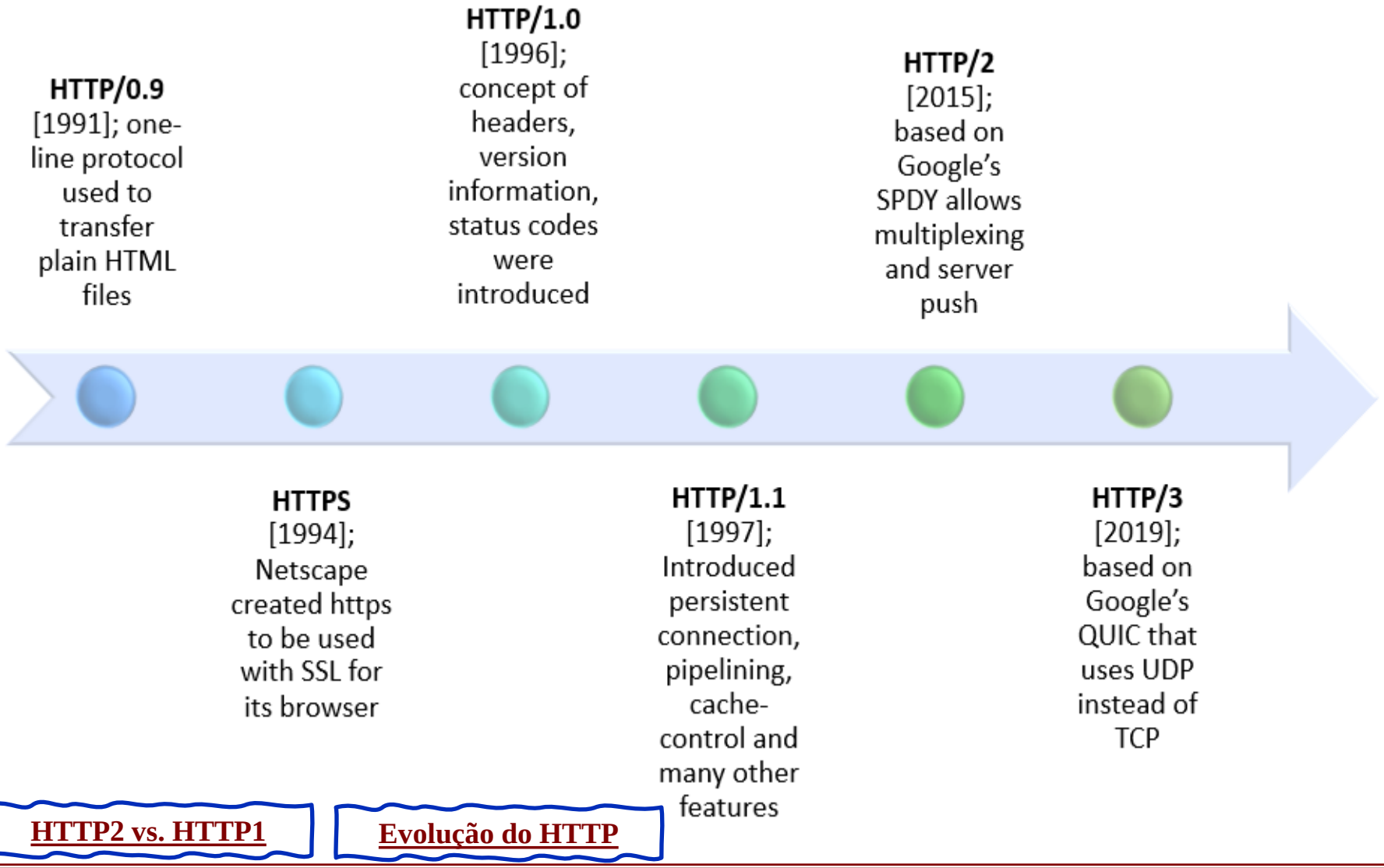


HTTP: Arquitetura LAMP



LAMP/WAMP

HTPP: Evolução

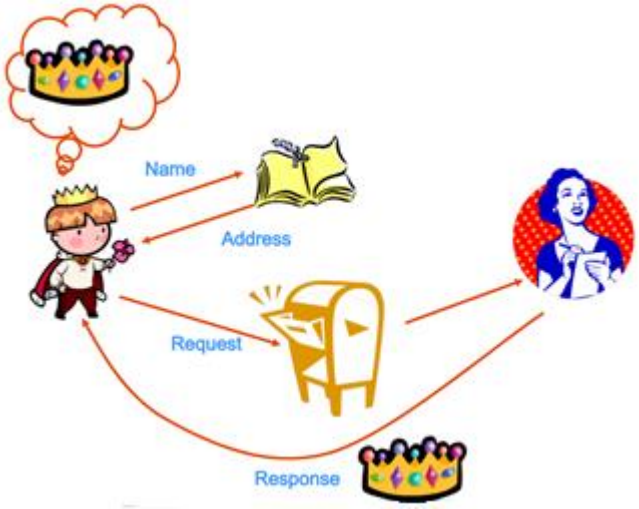


HTTP2 vs. HTTP1

Evolução do HTTP

HTTP: Funcionamento

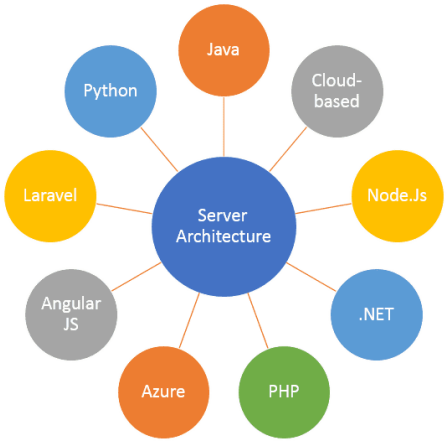
- HTTP: RFC 2616
- Modelo de funcionamento
  - pedido/resposta



HTTP: Servidor *Web*

- O servidor *Web* espera ligações no porto 80
  - O utilizador indica no navegador o URL de acesso

    Ex: http://localhost:8080 (desenvolvimento)



- O servidor *Web* não mantém estado (*stateless*, idempotente)
  - Simplifica a implementação e reduz os recursos de execução
  - Se ocorrer uma falha, o servidor pode ser reiniciado sem precisar de mecanismos elaborados de recuperação
  - Têm de ser utilizados mecanismos externos para manter o estado
  - Um servidor *Web* não suporta nativamente a execução de aplicações *Web* porque não tem competências para executar código

Best Open Source Web Servers

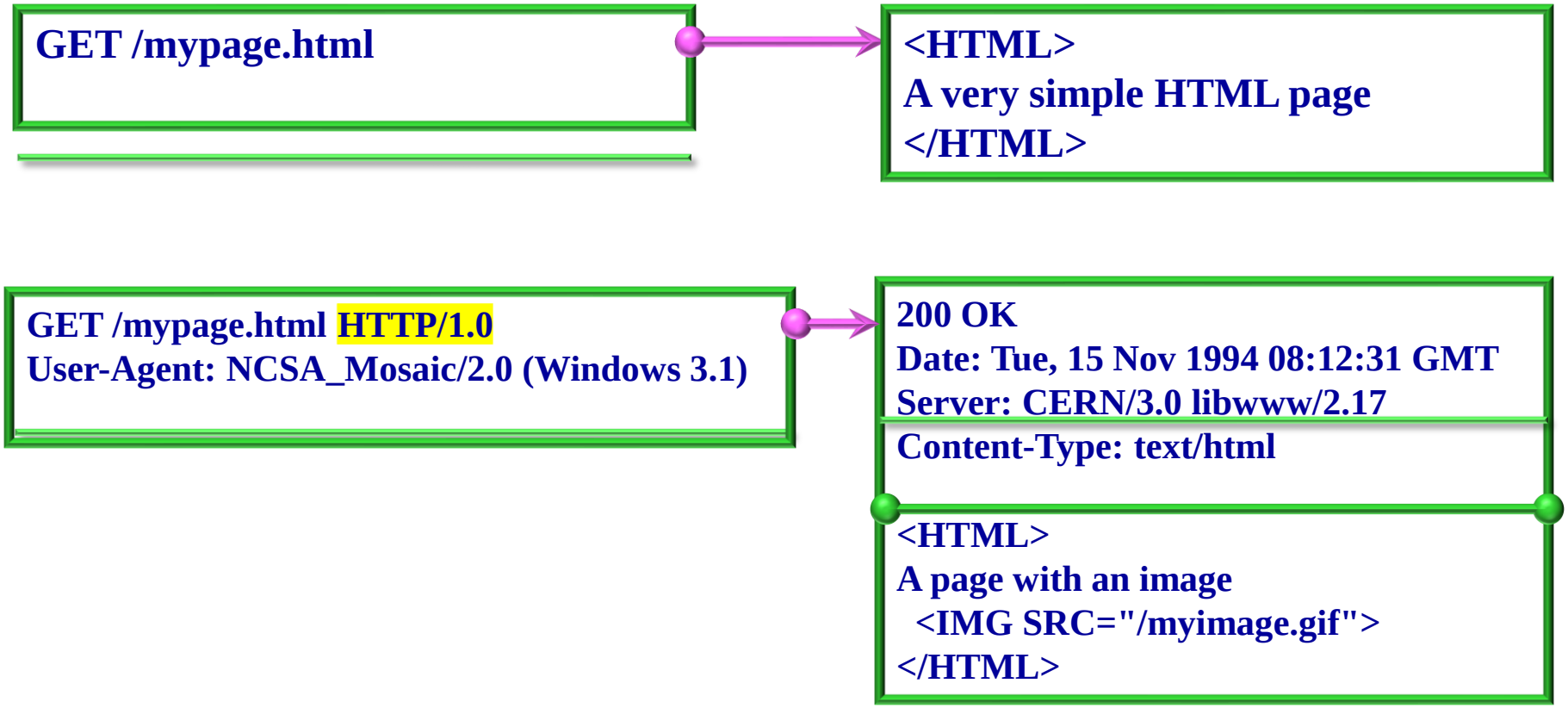
Comparison of Web Server Software

HTTP: Interação

- Interação Cliente *Web* (navegador/browser) ⟷ Servidor *Web*
  - O cliente interpreta o URL e obtém do DNS o endereço IP
  - O cliente estabelece ligação TCP (circuito virtual) com o servidor
  - O cliente, através do protocolo HTTP, envia um pedido referindo o recurso (ex. documento HTML, imagem) solicitado
  - O cliente recebe o recurso pedido ou página HTML com o erro
  - O cliente deteta que a ligação foi terminada pelo servidor
  - O cliente apresenta ao utilizador o recurso recebido

- Interação Cliente *Web* (navegador/browser) ⟵ Servidor *Web*
  - O servidor, por omissão, espera ligações TCP no porto 80
  - O servidor aceita ligação (circuito virtual) TCP
  - O servidor recebe o pedido do cliente que refere o recurso solicitado
  - O servidor após encontrar o recurso solicitado envia-o para o cliente se não encontrar envia uma página HTML com o erro
  - O servidor termina a ligação

HTTP: Exemplos de Mensagens



HTTP: Exemplos de Mensagens

```
GET /en-US/docs/Glossary/Simple_header HTTP/1.1
Host: developer.mozilla.org
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:50.0) Gecko/20100101 Firefox/50.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://developer.mozilla.org/en-US/docs/Glossary/Simple_header
```

```
200 OK
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Wed, 20 Jul 2016 10:55:30 GMT
Etag: "547fa7e369ef56031dd3bff2ace9fc0832eb251a"
Keep-Alive: timeout=5, max=1000
Last-Modified: Tue, 19 Jul 2016 00:59:33 GMT
Server: Apache
Transfer-Encoding: chunked
Vary: Cookie, Accept-Encoding
```

(content)

HTTP: Exemplos de Mensagens

```
GET /static/img/header-background.png HTTP/1.1
Host: developer.mozilla.org
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:50.0) Gecko/20100101 Firefox/50.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://developer.mozilla.org/en-US/docs/Glossary/Simple_header
```

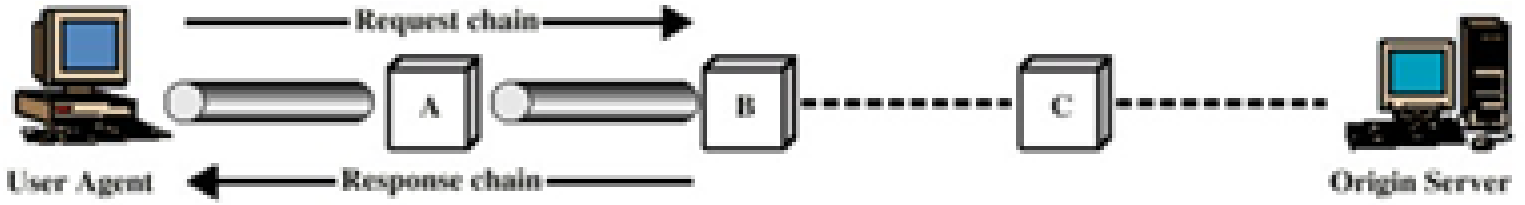
```
200 OK
Age: 9578461
Cache-Control: public, max-age=315360000
Connection: keep-alive
Content-Length: 3077
Content-Type: image/png
Date: Thu, 31 Mar 2016 13:34:46 GMT
Last-Modified: Wed, 21 Oct 2015 18:27:50 GMT
Server: Apache
```

```
(image content of 3077 bytes)
```

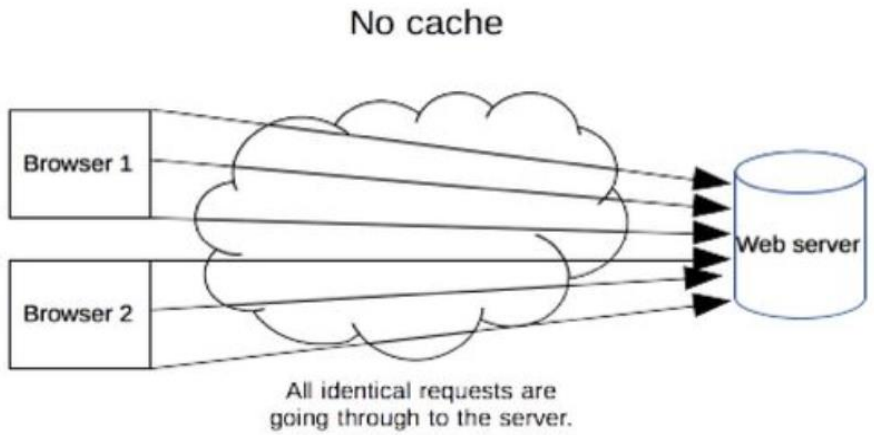
HTTP: Suporte Tecnológico

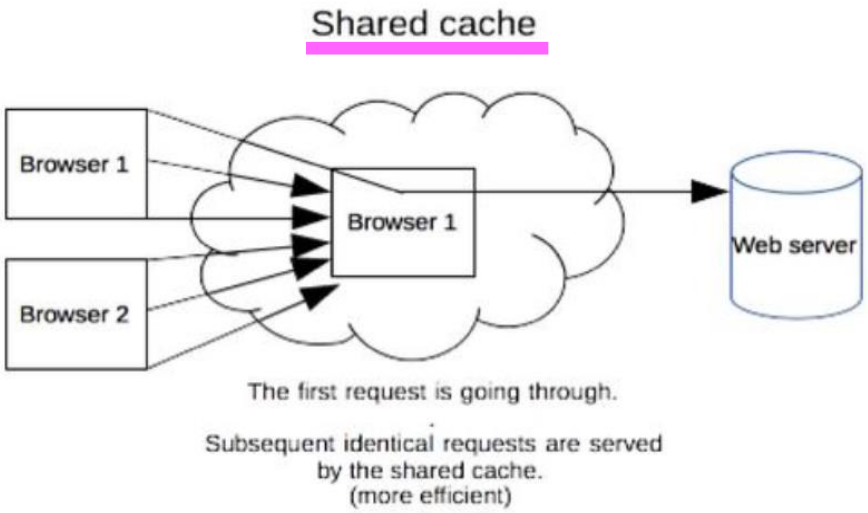
HTTP: Componentes da Arquitetura *Web*

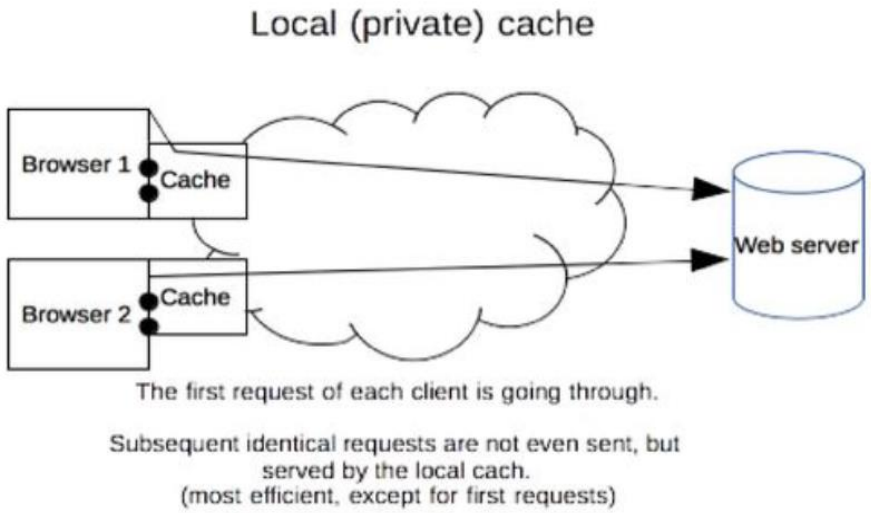
- Para além dos componentes mais comuns, isto é, clientes (navegadores) e servidores, existem outros que integram a arquitetura da *Web*, nomeadamente:
  - **Agente**: cliente que faz pedidos HTTP automatizados
  - ***Cache***: manutenção, perto dos navegadores, de cópias dos recursos mais requisitados para aumentar o desempenho



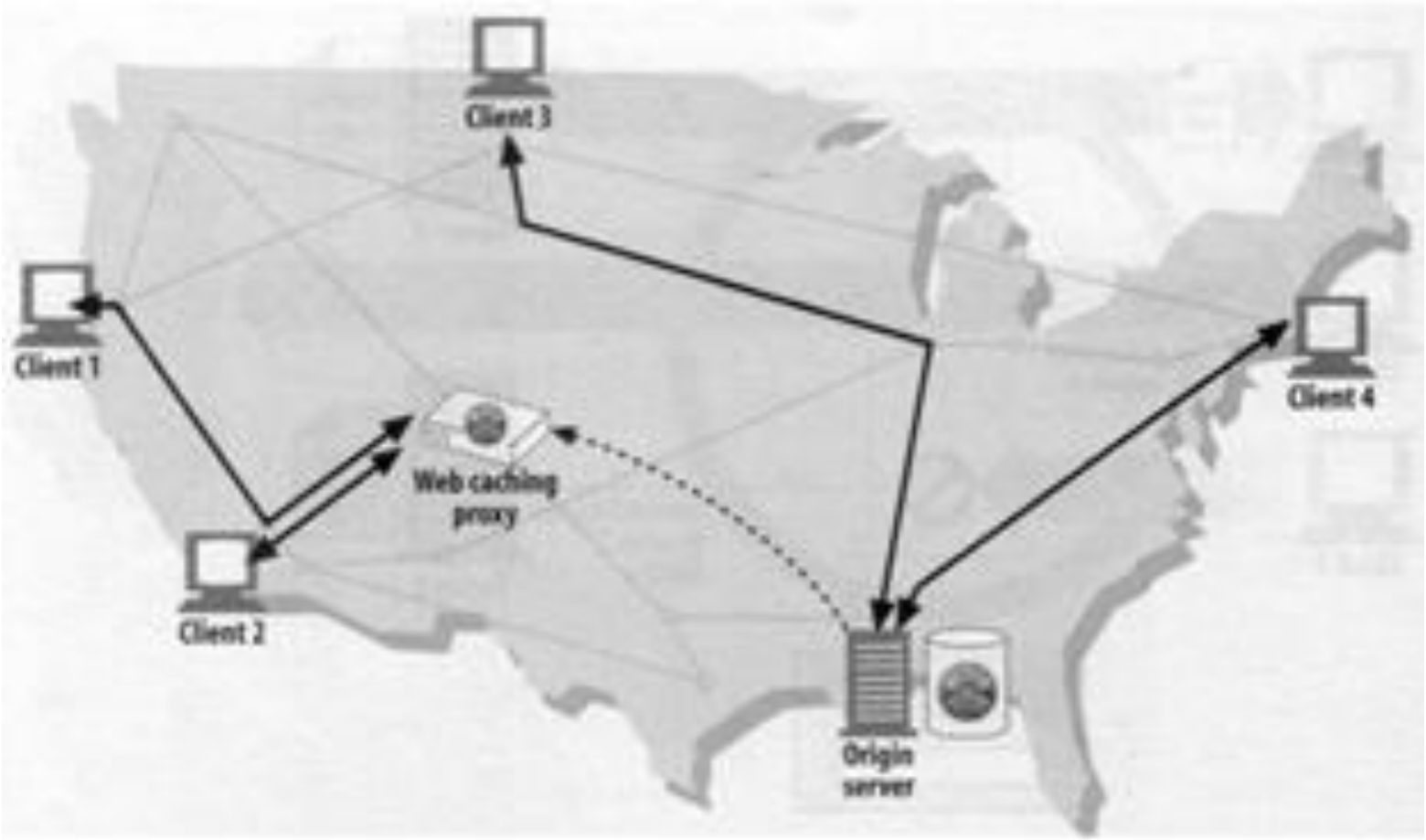
HTTP: Tipos de *Cache*







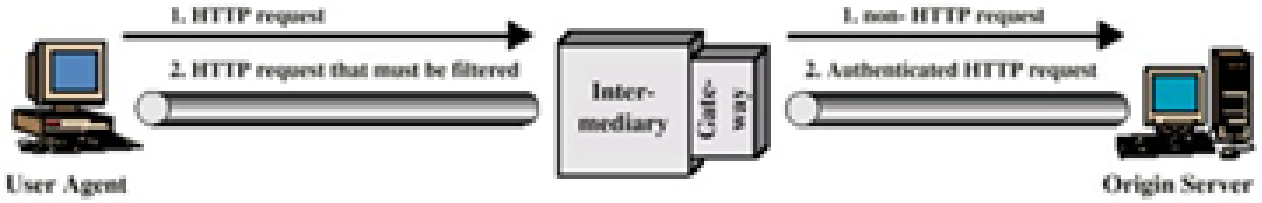
HTTP: *Web Cache*



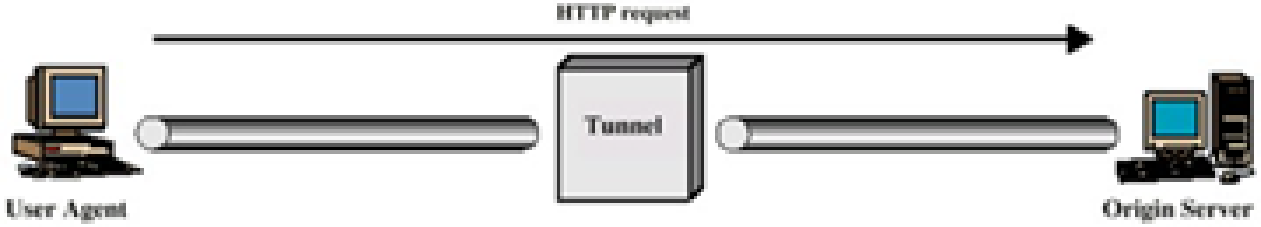
HTTP: Intermediários *Web*

- ***Proxy***: intermediário entre cliente e servidor

- ***Gateway***: servidor que interage com outros servidores

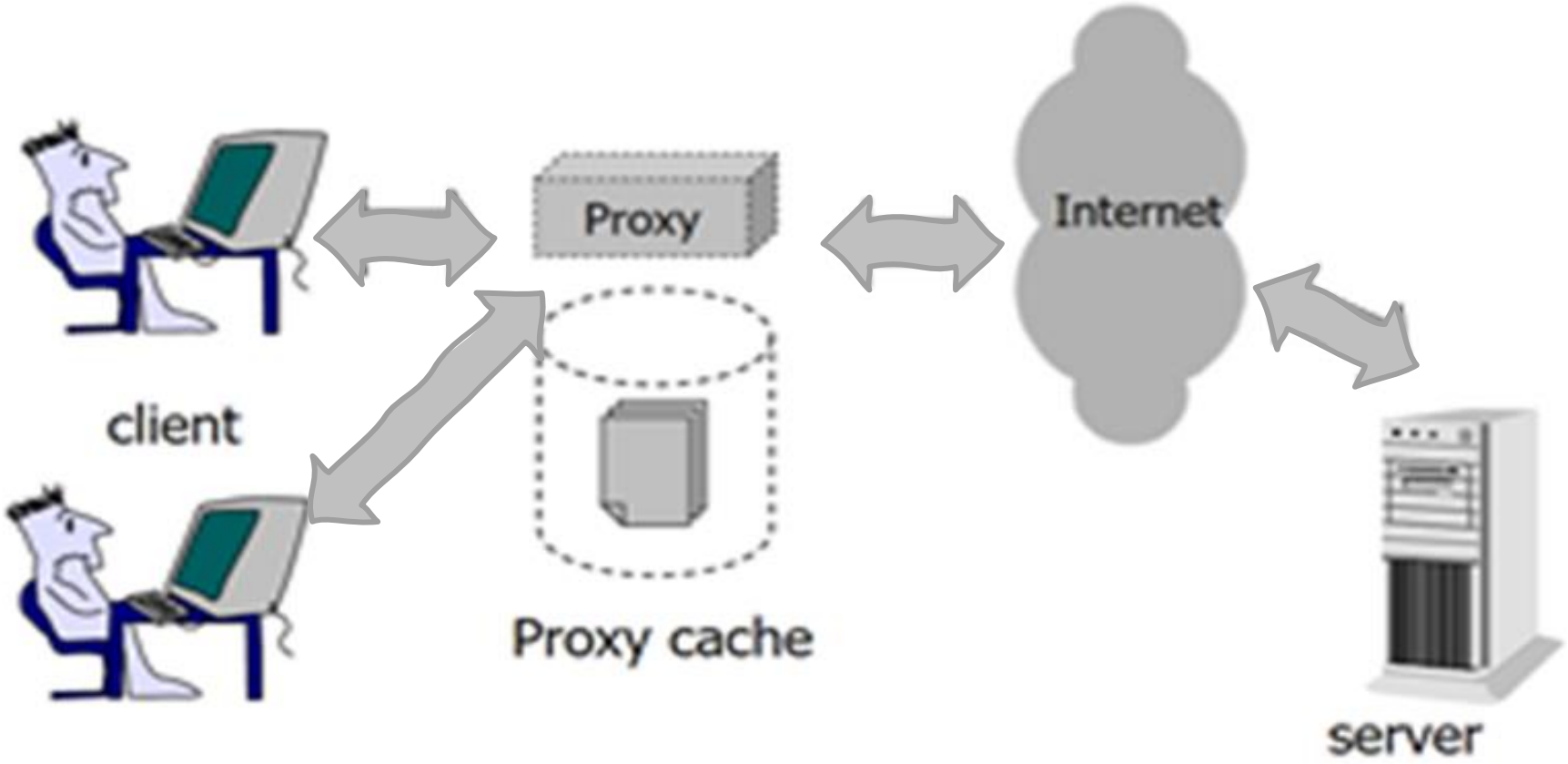


- ***Tunnel***: *proxy* que encaminha mensagens HTTP

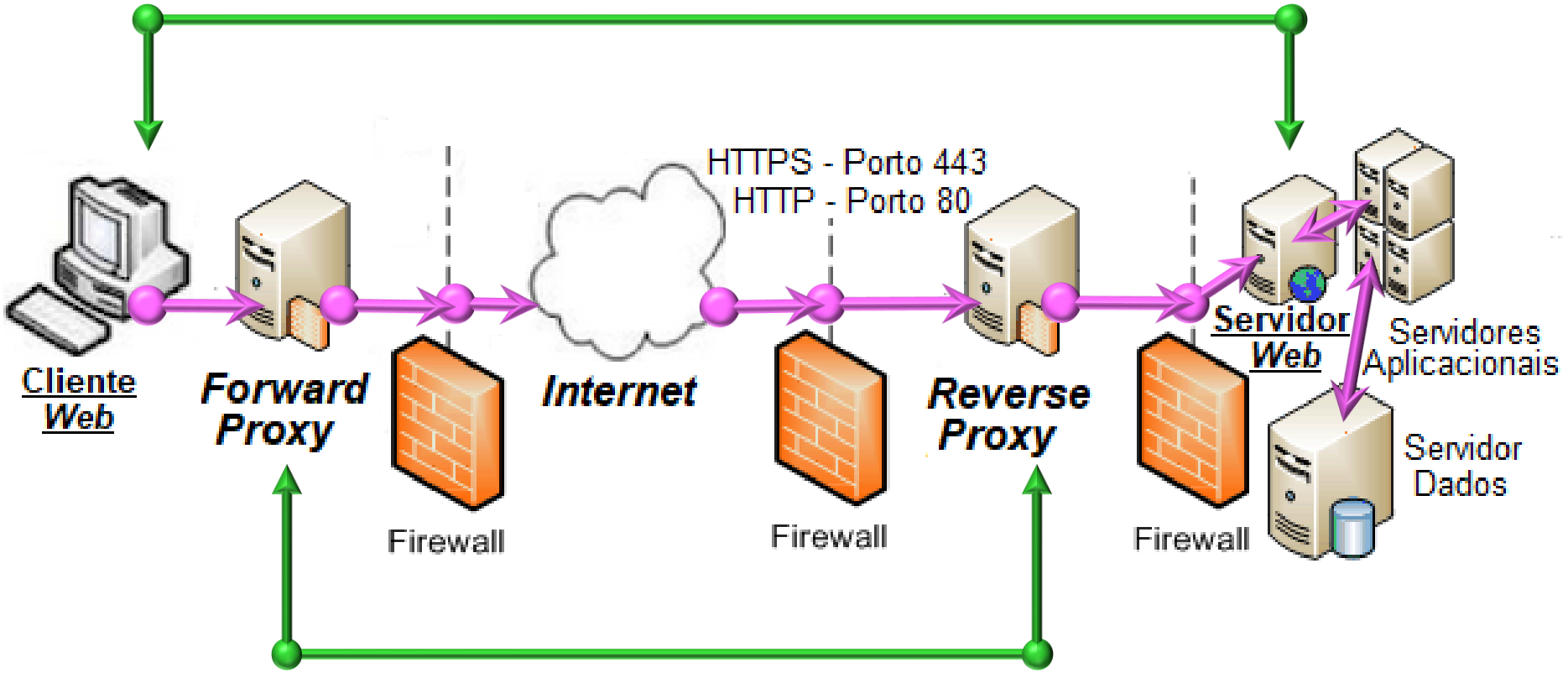


HTTP: *Proxy*

- *Proxy*: pode ser visto como um agente despachante, que envia/recebe mensagens associadas a um dado URL, rescreve parte ou a totalidade e reencaminha-as

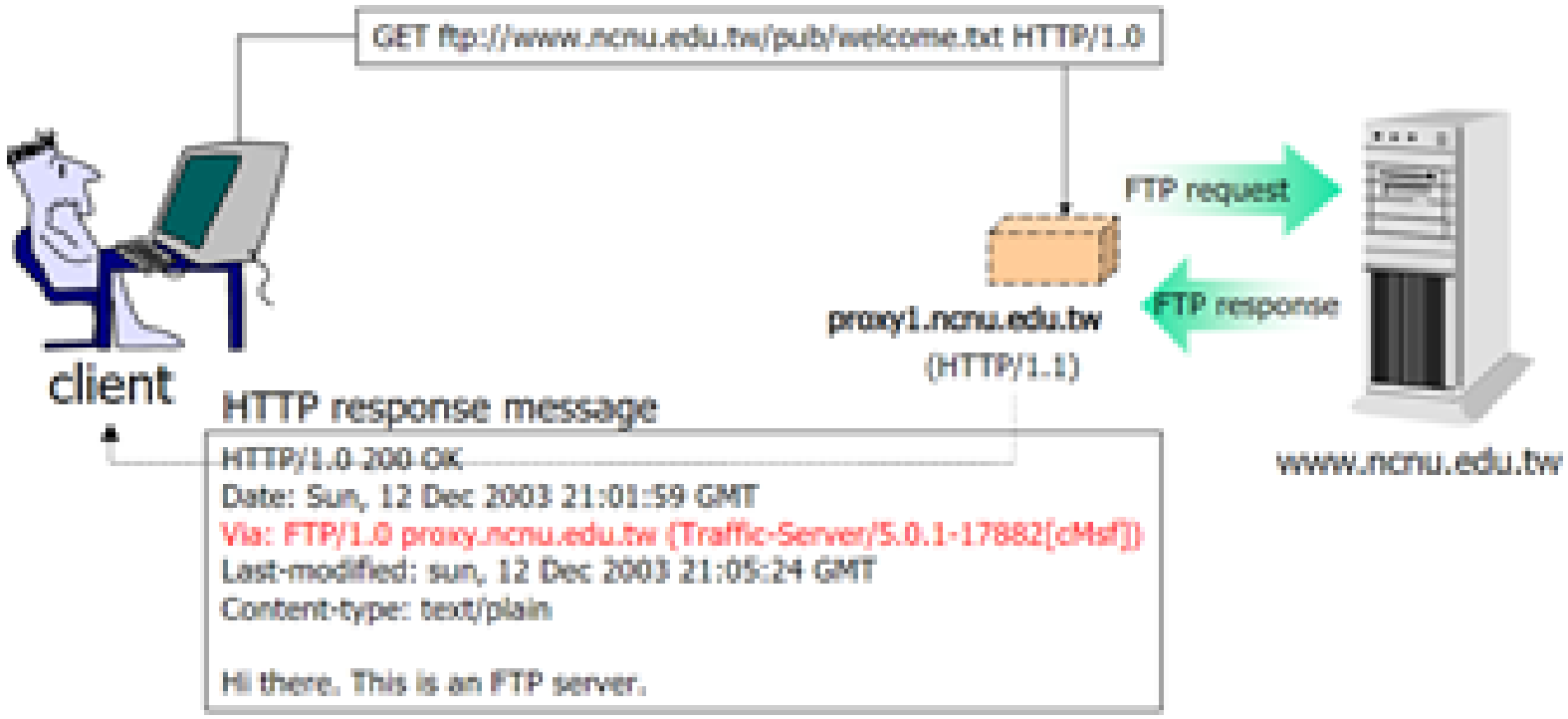


HTTP: *Forward & Reverse Proxy*

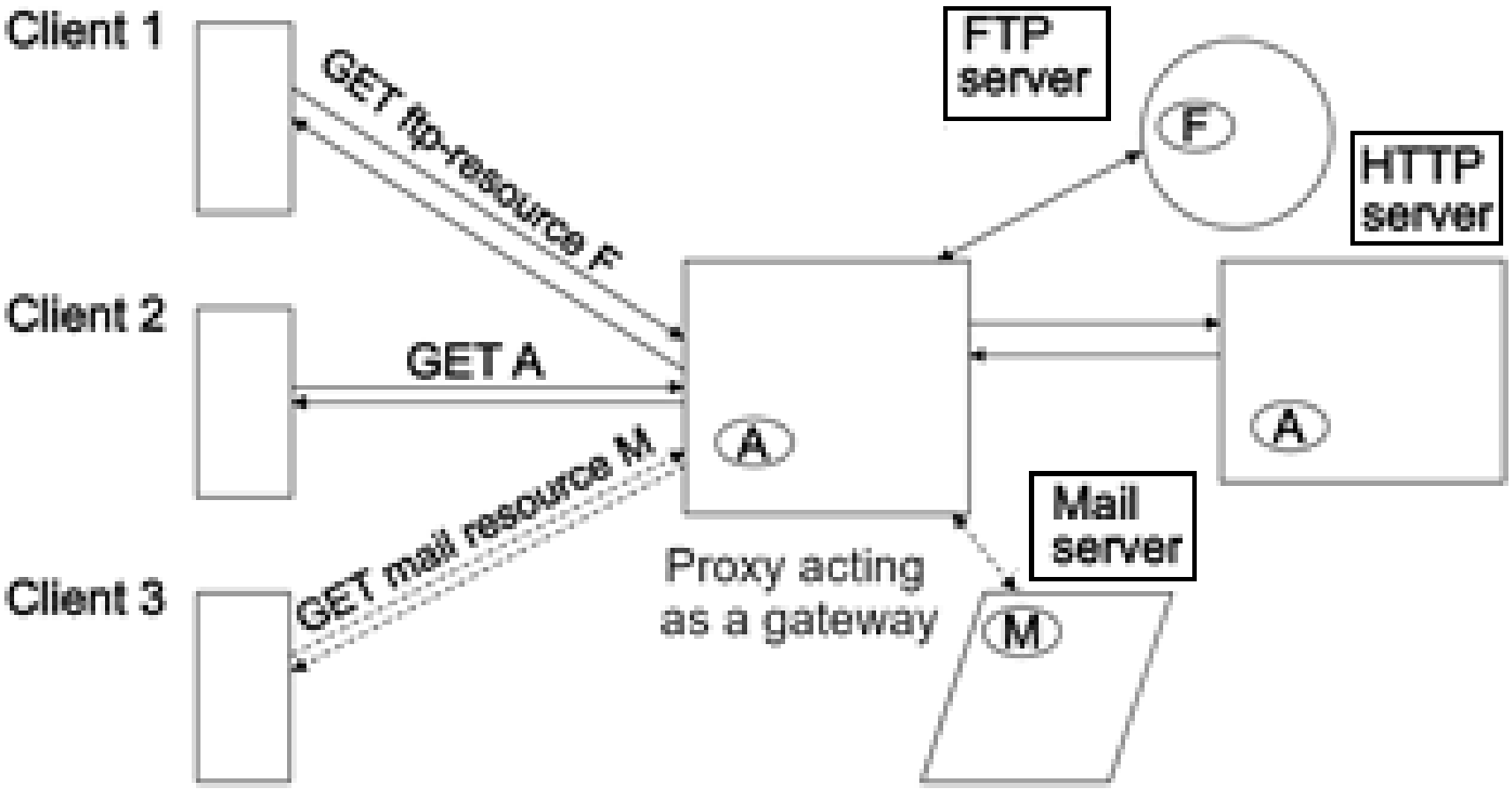


HTTP: *Gateway*

- *Gateway*: intermediário que permite transferir dados entre servidores. Se necessário converte pedidos para o protocolo do servidor de destino, basicamente é um conversor de protocolos

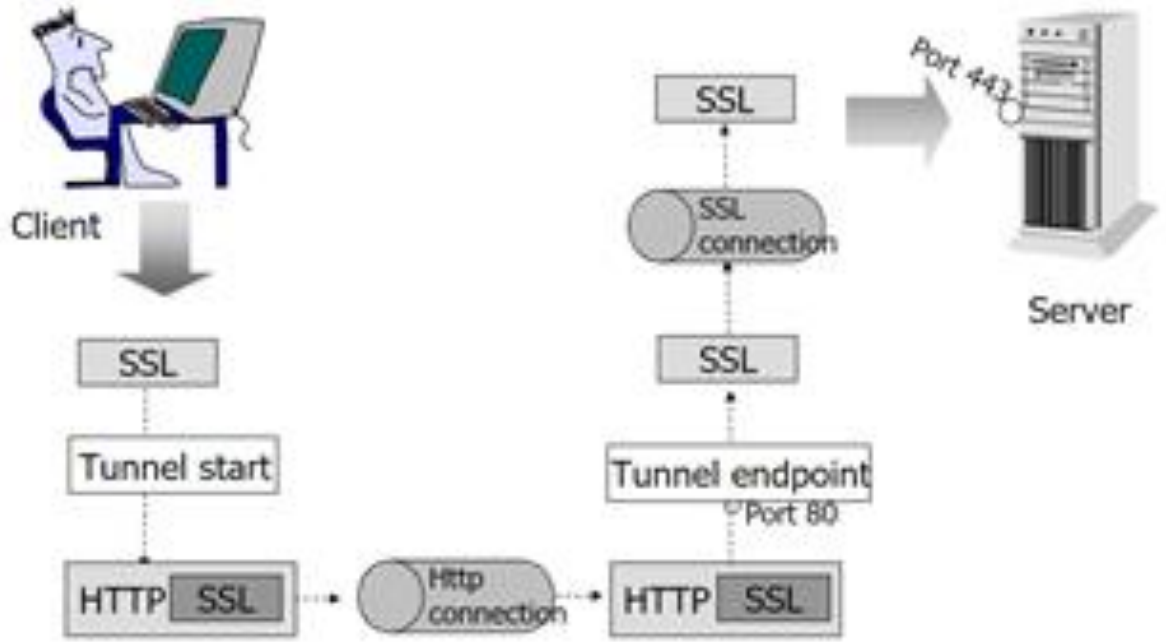


HTTP: *Gateway (HTTP, FTP, eMAIL)*



HTTP: *Tunnel*

- *Tunnel*: Funciona como interligação entre dois interlocutores que trocam dados, em cima de HTTP, entre si
  - Cliente e ou servidor com *firewall*
  - Meio de interligação não confiável (*Internet*)
  - Estabelecimento de ligações cifradas



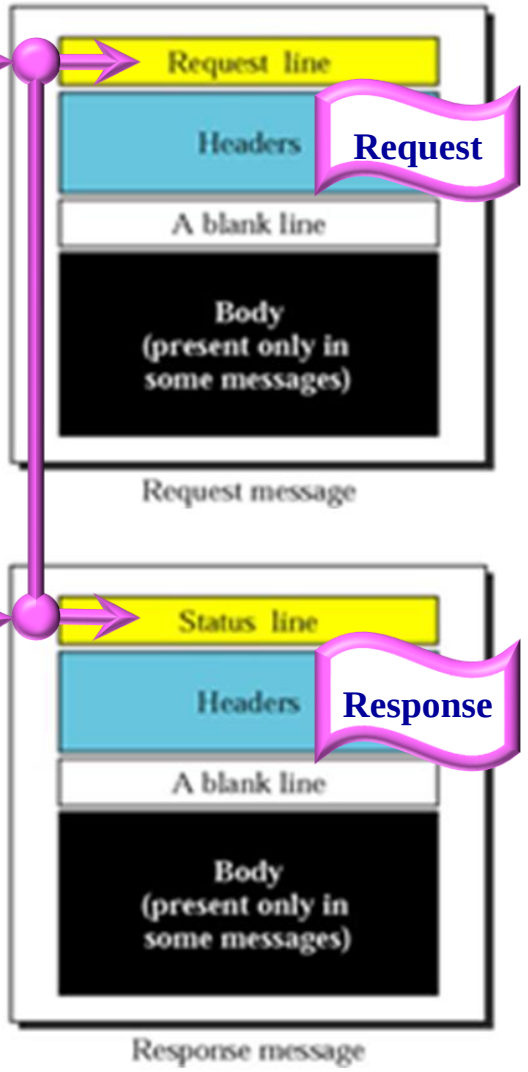
HTTP: Pedidos e Respostas

- O pedido do cliente ao servidor e a respetiva resposta são mensagens especificadas no protocolo HTTP

- As mensagens HTTP incluem três componentes:
  - Linha Inicial
    - **Cliente → Servidor:** indica o que se quer fazer com o pedido
    - **Cliente ← Servidor:** indica o que aconteceu com o processamento do pedido
  - Cabeçalhos
    - **Cliente → Servidor:** indica a versão do navegador, os formatos dos recursos que reconhece, etc…
    - **Cliente ← Servidor:** indica a versão do servidor, dimensão dos recursos, data última modificação, etc…
  - Corpo
    - Parte opcional da mensagem contém dados (texto ou binário)

HTTP: Mensagens

- **Pedido (cliente → servidor)**
  - A primeira linha
    - Identificação do pedido, denominado método
    - A localização do recurso em causa (URL)
    - A versão do protocolo HTTP
  - As linhas seguintes dependem do método indicado e contêm os cabeçalhos (RFC 2822)
  - A última linha é obrigatoriamente vazia (CRLF)
- **Resposta (cliente ← servidor)**
  - A primeira linha
    - Indica o estado do processamento, se teve sucesso ou, caso contrário, o código de erro e respetiva página HTML
  - As linhas seguintes contêm cabeçalhos (RFC 2822)
  - Seguidamente vêm os dados da resposta (ex: página HTML ou imagem)



HTTP: Tipos de Conteudo

- As mensagens HTTP usam o formato MIME *Multipurpose Internet Mail Extensions* para especificar o conteúdo

- O formato MIME é um rótulo que refere um tipo e um subtipo
  - Texto HTML: texto/html
  - Texto ASCII: texto/plain
  - Imagem no formato JPEG: image/jpeg
  - Imagem no formato GIF: image/gif
  - Filme no formato QuickTime: vídeo/quicktime
  - Microsoft PowerPoint: application/vnd.ms-powerpoint

Tipos MIME (IANA) **Media Type Specifications and Registration Procedures**

HTTP: Sintaxe das Mensagens

generic-message ::= <start-line> *(<message-header> CRLF) CRLF [< message-body>]

start-line ::= <request-line> | <status-line>

message-header ::= header-name**:**header-value CRLF

request-line ::= <method> SP <request-URI> SP <HTTP-Version> CRLF

method ::= "OPTIONS" | "**GET**" | "**HEAD**" | "**POST**" | "PUT"
| "DELETE" | "TRACE" | "CONNECT" | <extension-method>

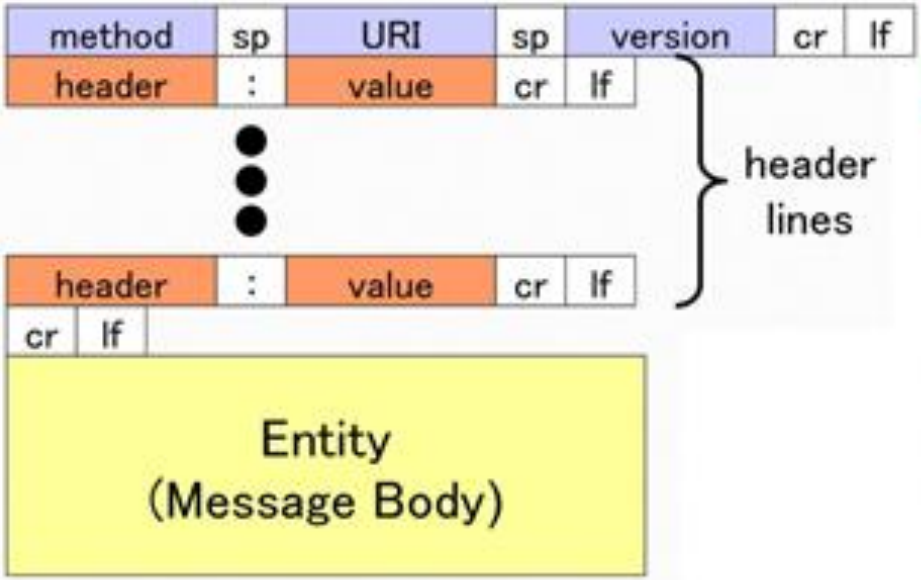
request-URI ::= "*" | <absoluteURI> | <absPath>

response ::= <status-line> *(general-header | response-header | entity-header) CRLF
[message-body]

status-line ::= <HTTP-Version> SP <status-code> SP <reason-phrase> CRLF

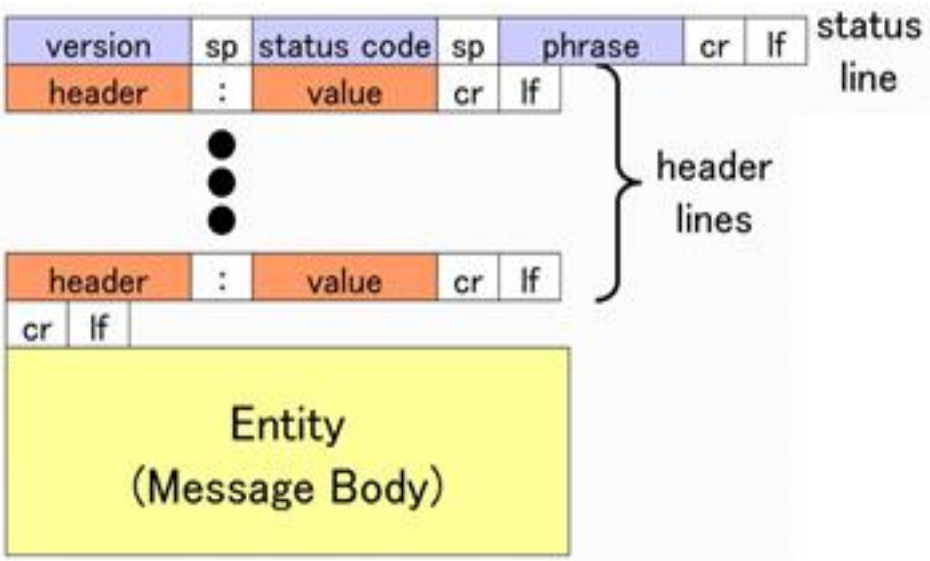
Hypertext Transfer Protocol -- HTTP/1.1

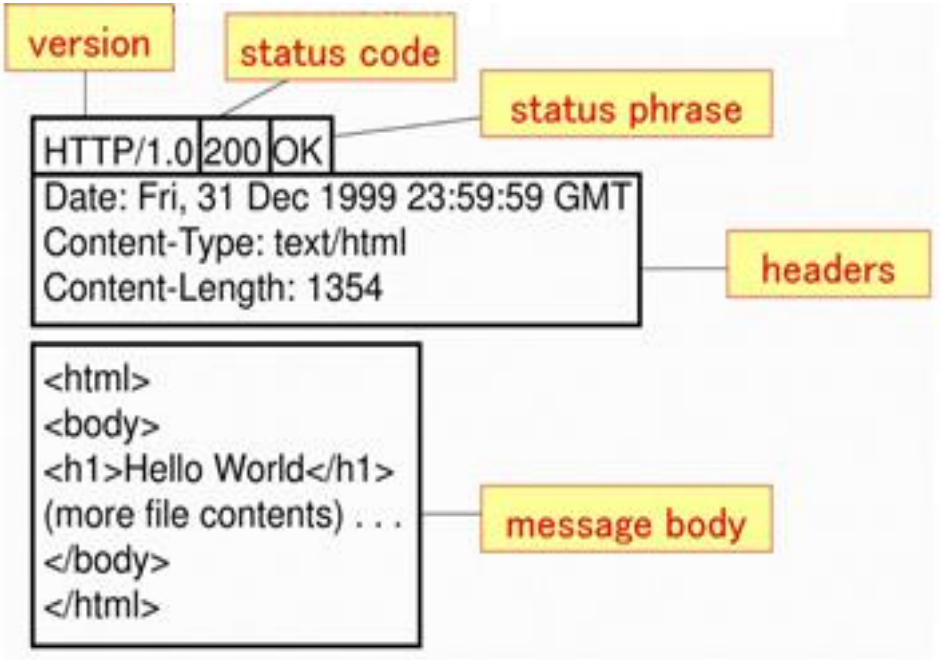
HTTP: Mensagem de Pedido



The URI is specified without the host name, unless the request is sent to a proxy

HTTP: Mensagem de Resposta





HTTP: Códigos de Estado

Hypertext Transfer Protocol (HTTP) Status Code Registry

HTTP & Gatos

HTTP: Códigos de Estado mais Comuns

200 OK
201 Criado
204 Sem conteúdo
304 Não modificado
400 Pedido mal formado
401 Não tem autorização
404 Não encontrado
501 Comando não implementado
503 Serviço não dispoenível

HTTP Status Messages

HTTP: Cabeçalhos

- Os cabeçalhos permitem a troca de informação entre cliente e servidor

- **Quatro categorias**
  - ***Geral*** – Informação que não está diretamente relacionada com o cliente, o servidor e o HTTP
  - ***Pedido*** – Permite indicar preferências nos formatos a receber e parâmetros do servidor
  - ***Resposta*** – Informação providenciada pelo servidor
  - ***Dados*** – Informações referentes aos dados transferidos

- **Cabeçalhos gerais**
  - ***Cache-control*** – Indica como efetuar a *cache* das páginas no cliente (sim, não e tempo de validade)
  - ***Upgrade*** – Indica a versão do protocolo HTTP a usar
  - ***MIME-version*** – Indica a versão do MIME a utilizar na comunicação
  - ***Date*** – Formato preferencial para a data

HTTP: Cabeçalhos

- **Cabeçalhos do pedido** (cliente ⟶ servidor)
  - ***Accept*** – Indica os tipos de dados que sabe interpretar (ex: */*, image/jpeg, text/html, application/msword, etc…)
  - ***Cookie*** – Permite associar um estado ao cliente usando um par atributo valor
  - ***If-modified-since*** – indica data e hora da ultima atualização do recurso

- **Cabeçalhos da resposta** (cliente ⟵ servidor)
  - ***Set-cookie*** – Fornece um par atributo valor que fica associado ao cliente que permite a manutenção de estado
  - ***Server*** – Informa qual o servidor que está a processar o pedido e qual a sua versão

- **Cabeçalhos de dados:**
  - ***Content-length*** – Dimensão em *bytes* do corpo de dados
  - ***Content-type*** – Tipo (MIME) de dados do corpo (ex: image/jpeg, text/html, etc…)
  - ***Expires*** – Indica até quando o recurso provavelmente não será alterado
  - ***Last-modified*** – Data e hora da última modificação
  - ***Location*** – Indica que o recurso foi movido para noutro URL

Cabeçalhos HTTP

Resumo: Cabeçalhos & Códigos Estado

General headers

Header	Description
Cache-control	Specifies information about caching
Connection	Shows whether the connection should be closed or not
Date	Shows the current date
MIME-version	Shows the MIME version used
Upgrade	Specifies the preferred communication protocol

Request headers

Header	Description
Accept	Shows the media format the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	Shows what permissions the client has
From	Shows the e-mail address of the user
Host	Shows the host and port number of the client
If-modified-since	Send the document if newer than specified date
If-match	Send the document only if it matches given tag
If-non-match	Send the document only if it does not match given tag
If-range	Send only the portion of the document that is missing
If-unmodified-since	Send the document if not changed since specified date
Referrer	Specifies the URL of the linked document
User-agent	Identifies the client program

Response headers

Header	Description
Accept-range	Shows if server accepts the range requested by client
Age	Shows the age of the document
Public	Shows the supported list of methods
Retry-after	Specifies the date after which the server is available
Server	Shows the server name and version number

Entity headers

Header	Description
Allow	Lists valid methods that can be used with a URL
Content-encoding	Specifies the encoding scheme
Content-language	Specifies the language
Content-length	Shows the length of the document
Content-range	Specifies the range of the document
Content-type	Specifies the media type
Etag	Gives an entity tag
Expires	Gives the date and time when contents may change
Last-modified	Gives the date and time of the last change
Location	Specifies the location of the created or moved document

Status codes

Code	Phrase	Description
	Informational	
100	Continue	The initial part of the request has been received and the client may continue with its request.
101	Switching	The server is complying with a client request to switch protocols defined in the upgrade header.
	Success	
200	OK	The request is successful.
201	Created	A new URL is created.
202	Accepted	The request is accepted, but it is not immediately acted upon.
204	No content	There is no content in the body.
	Redirection	
301	Multiple choices	The requested URL refers to more than one resource.
302	Moved permanently	The requested URL is no longer used by the server.
304	Moved temporarily	The requested URL has moved temporarily.
	Client Error	
400	Bad request	There is a syntax error in the request.
401	Unauthorized	The request lacks proper authorization.
403	Forbidden	Service is denied.
404	Not found	The document is not found.
405	Method not allowed	The method is not supported in this URL.
406	Not acceptable	The format requested is not acceptable.
	Server Error	
500	Internal server error	There is an error, such as a crash, at the server site.
501	Not implemented	The action requested cannot be performed.
503	Service unavailable	The service is temporarily unavailable, but may be requested in the future.

HTTP: Recurso Original

- A fresh copy is obtained from the origin server if the request includes the following header
  - `Cache-Control: no-cache`
- The proxy must revalidate its copy with the origin server if the following header is included in the request
  - `Cache-Control: max-age=0`

HTTP: Métodos Comuns

- Método **GET**
  - Pede um recurso ao servidor indicando um URL
    (ex: ficheiro, CGI, etc...)
  - A resposta do servidor contém:
    - Na primeira linha, o estado
    - Nas seguintes os cabeçalhos
    - E em seguida o recurso pedido
- Método **HEAD**
  - Semelhante ao GET, mas a resposta não inclui o recurso
  - Utilizado para saber informações sobre um recurso
  - Útil para saber:
    - Dados sobre a *cache* do *browser*
    - Averiguar o tempo de *download* do recurso
- Método **POST**
  - Semelhante ao GET, mas atua silenciosamente
  - Enviar dados no corpo da mensagem

HTTP: Exemplo GET (via telnet)

$telnet phoenix 80

C: Trying 10.64.11.121...
C: Connected to phoenix.
C: Escape character is '^]'.
C:
C: GET /exemplo.html HTTP/1.1
C: Host: phoenix
C:

Indica fim do pedido

***S:** HTTP/1.1 200 OK*
***S:** Date: Mon, 26 Apr 2004 08:20:24 GMT*
***S:** Server: Apache/2.0.40 (Red Hat Linux)*
***S:** Last-Modified: Fri, 23 Apr 2004 16:47:08 GMT*
***S:** ETag: "15b738-1740-c73a3b00"*
***S:** Accept-Ranges: bytes*
***S:** Content-Length: 5952*
***S:** Content-Type: text/html; charset=ISO-8859-1*
S:

Linha de separação

***S:** <html>*
***S:** <head>*
***S:** <meta http-equiv="Content-Language" content="pt">*
***S:** <title>Exemplo: Phoenix [SES Server]</title>*
***S:** </head>*
***S:** <body >*

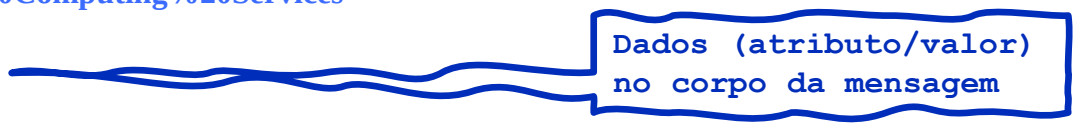
<h1>Saudações Académicas</h1>
…

***S:** </body>*
S: </html>
S:
C: Connection closed by foreign host.

HTTP: Exemplo POST (via telnet)

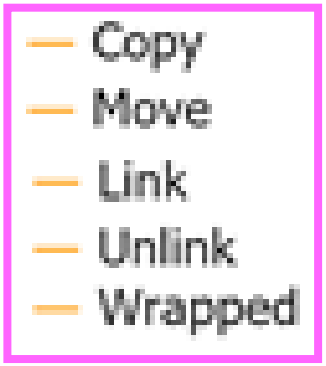
$telnet phoenix 80

C: POST /cgi-bin/post-query HTTP/1.1
C: Host: phoenix
C: Accept: */*
C: User-Agent: Lynx/2.2 libwww/2.14
C: From: grobe@www.cc.ukans.edu
C: Content-type: application/x-www-form-urlencoded
C: Content-length: 150
C:
C: org=Academic%20Computing%20Services
C: &users=10000
C: &browsers=lynx
C: &browsers=cello
C: &browsers=mosaic
C: &others=MacMosaic%2C%20WinMosaic
C:



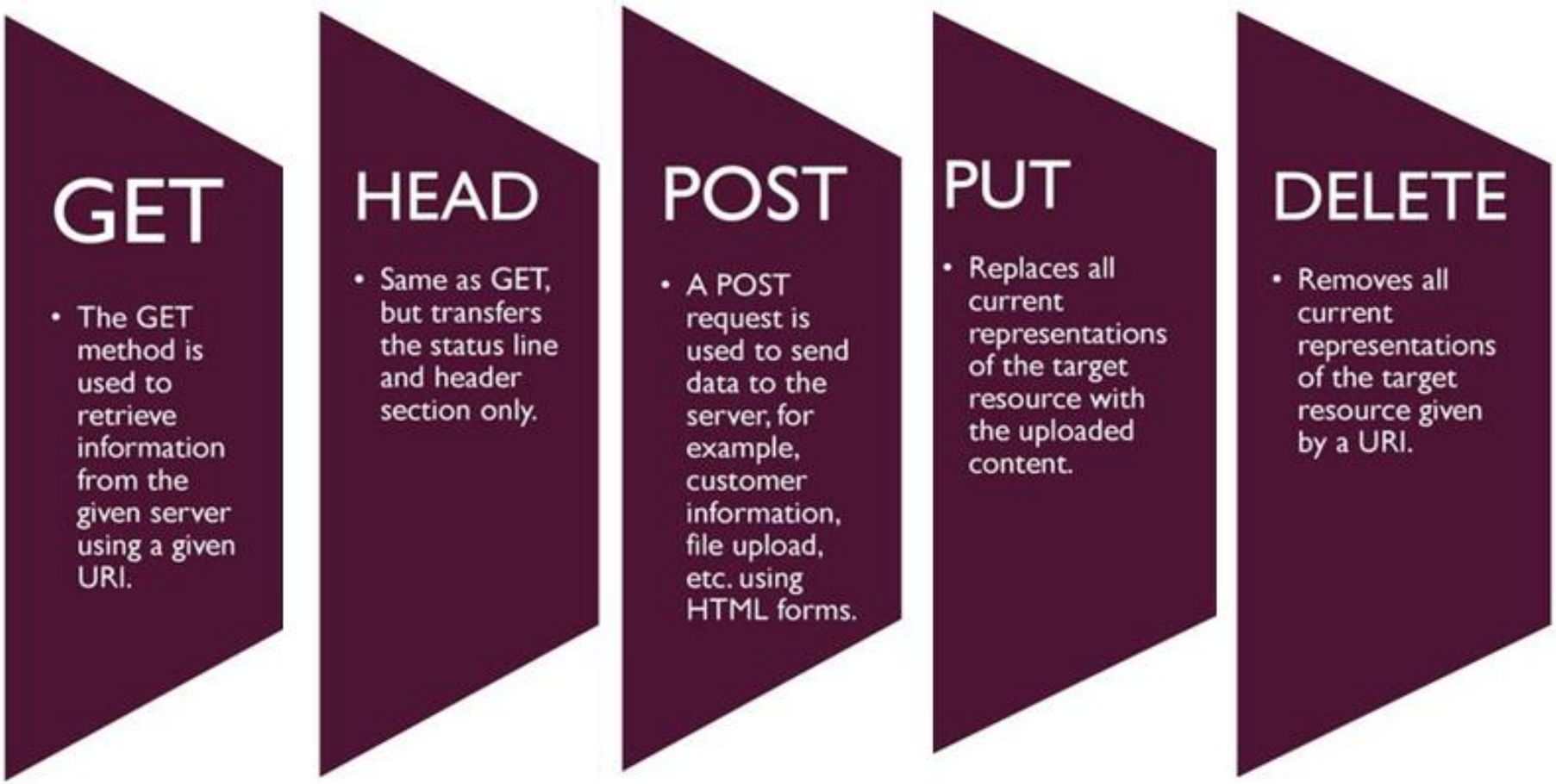
- ***S: …***
- ***S: Content-Type: text/html***
- ***S:***
- ***S: <h1>Query Results</h1>***
- ***S: You submitted the following name/value pairs:***
- ***S: ***
- ***S: org = Academic Computing Services ***
- ***S: users = 10000 ***
- ***S: browsers = cello ***
- ***S: browsers = lynx ***
- ***S: browsers = xmosaic ***
- ***S: others = Mac Mosaic, Win Mosaic ***
- ***S: ***

Resumo: Métodos ou Comandos



Command	Description
GET	Request for the resource located at the specified URL.
HEAD	It like GET, but without the body of response.
POST	Submits data to be processed to the identified resource. The data is included in the body of the request.
PUT	Uploads a representation of the specified resource.
DELETE	Deletes the resource located at the specified URL.
TRACE	Echoes back the received request, so that a client can see what (if any) changes or additions have been made by intermediate servers.
OPTIONS	Returns the HTTP methods that the server supports for specified URL. This can be used to check the functionality of a web server by requesting '*' instead of a specific resource.
CONNECT	Converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy.
PATCH	Is used to apply partial modifications to a resource.

HTTP: Métodos Comuns

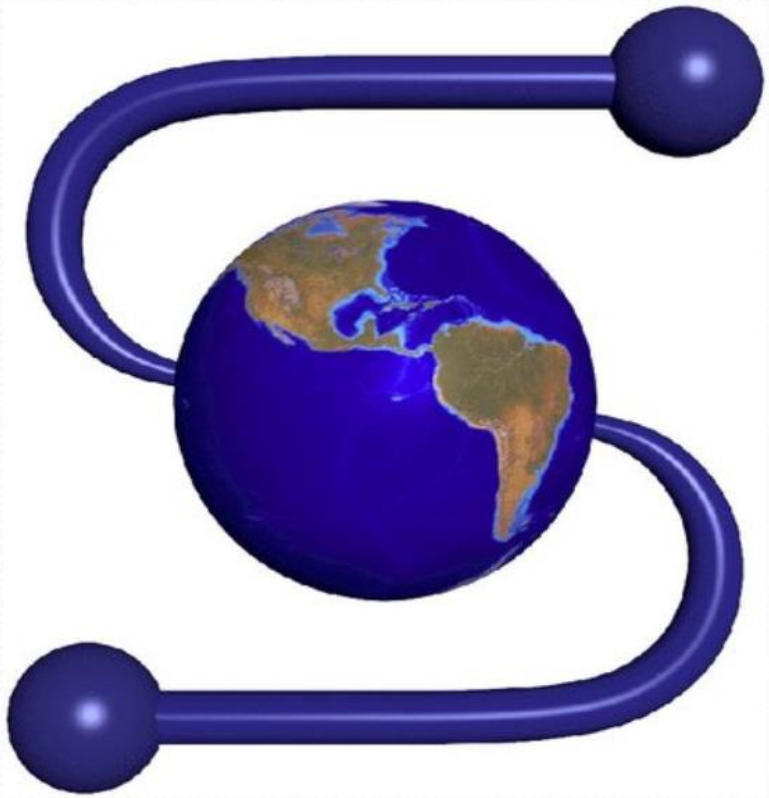


HTML

(Hyper Text Markup Language)

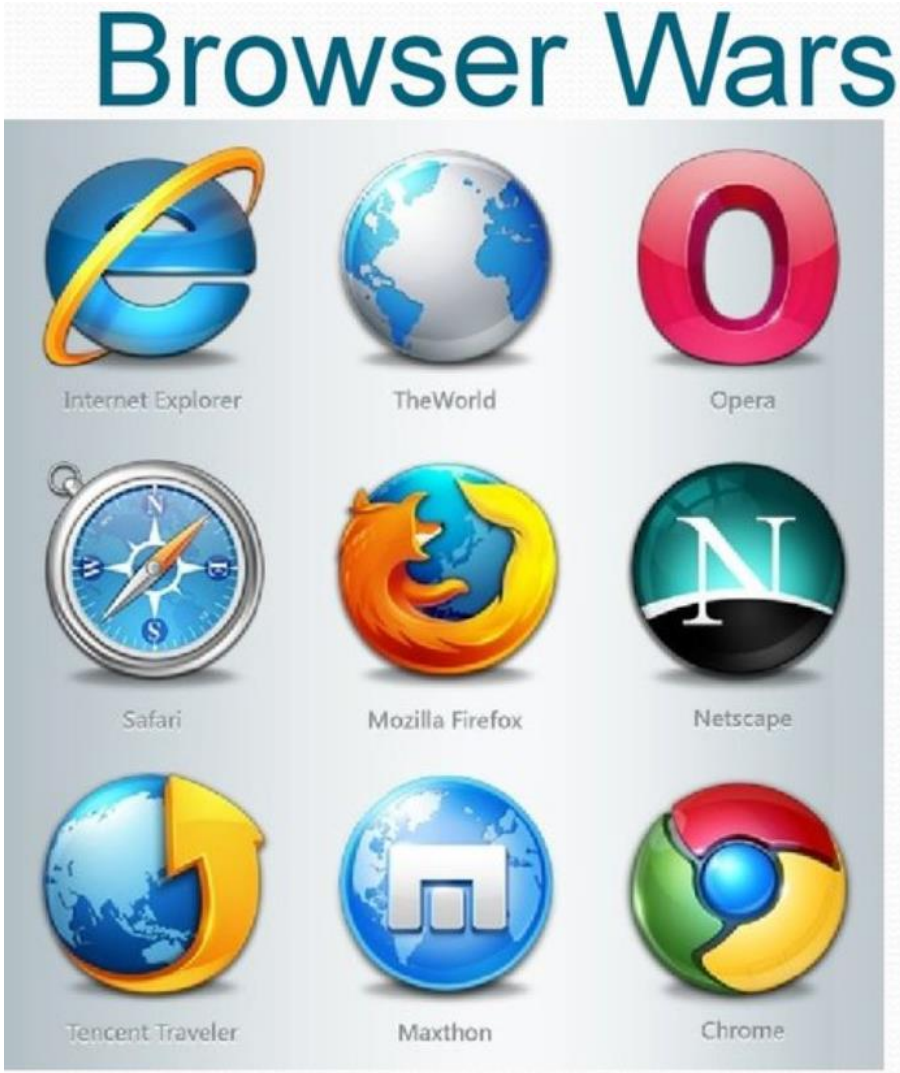
O Primeiro Navegador

Mosaic



- Released in 1993
- First Internet Web Browser
- First commercial software with graphical access to the content on the internet
- Very slow
- Did not handle loading pictures well

Evolução dos Navegadores



Compatibilidade dos Browsers
Caraterísticas do Browser

- Netscape was the standard until 1998.
- it folded and was taken over by AOL.
- Microsoft internet explorer snagged 96% of the browser market.
- IE has only recently been challenged by the Mozilla browser and Google chrome.

Navegadores em Portugal (2020)

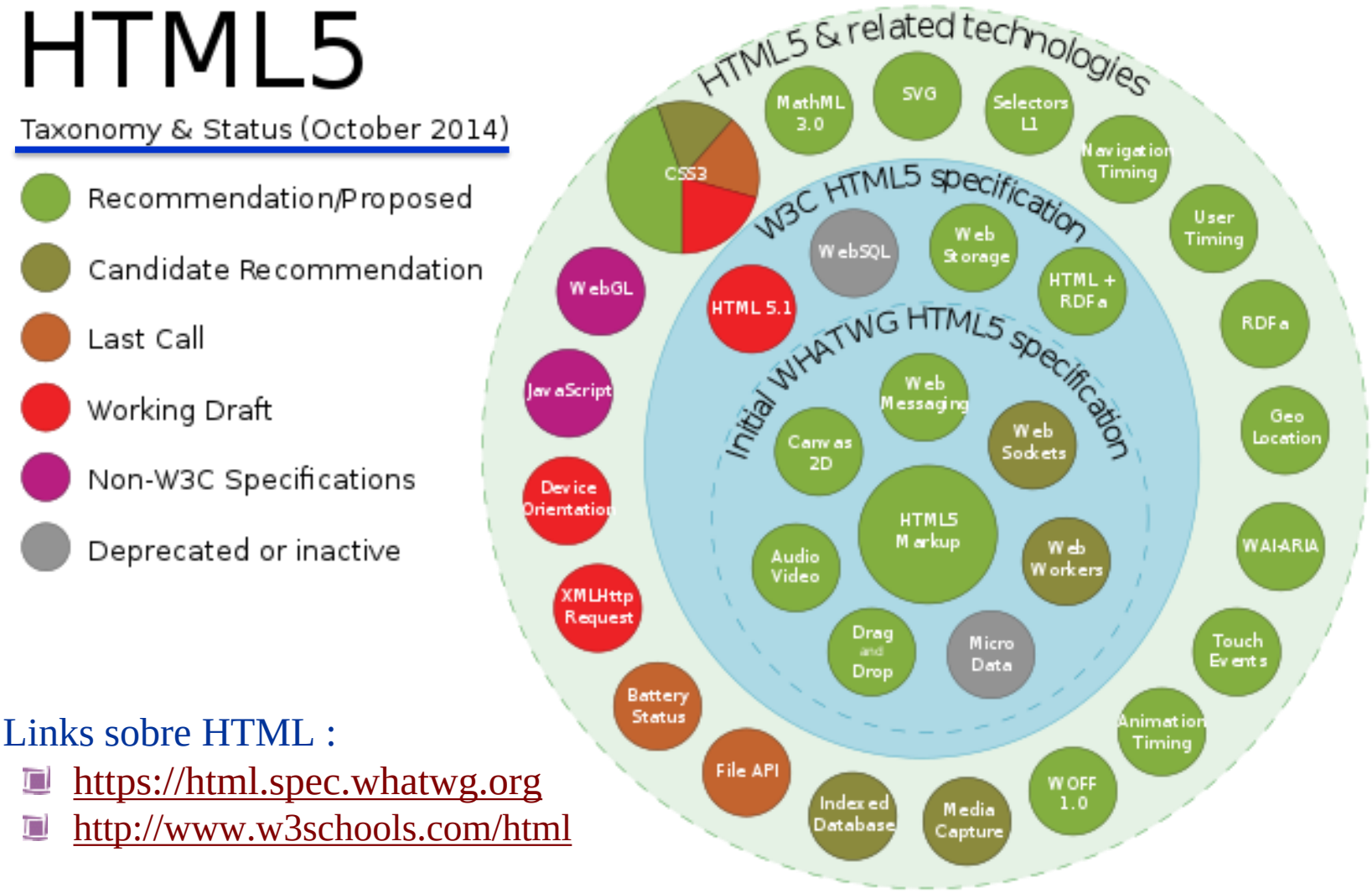
Logotipo	Navegador	Data de criação	Versão	Percentagem
	Google Chrome	2 de Setembro de 2008	90.0.4430.93 (26 de abril de 2021)	69,44%
	Safari	7 de Janeiro de 2003	14.0 (MacOS) (16 de setembro de 2020)	11,33%
	Mozilla Firefox	9 de Novembro de 2004	88.0 (19 de abril de 2021)	7,5%
	Internet Explorer	16 de Agosto de 1995	11.239.18362.0	4,08%
	Microsoft Edge	29 de Julho de 2015	90.0.818.46 (21 de abril de 2021)	3,39%

HyperText Markup Language

- Linguagem com origem na *Standard Generalized Markup Language* (SGML)
- Baseada em etiquetas "*tags*" que são interpretadas pelos navegadores "*browsers*" para apresentarem os recursos incluídos
- Permite descrever, independentemente da plataforma, dados e a forma como são apresentados
- O XHTML 1.0 é uma evolução da versão HTML 4.01 publicada em Dezembro de 1999
- O HTML 5 surge como um alternativa:

Version	First draft	Candidate recommendation	Recommendation
HTML5.0	2007	2012	2014
HTML5.1	2012	2015	2016
HTML5.2	2015	2017	2017
HTML5.3	2017	N/A	N/A

HyperText Markup Language 5



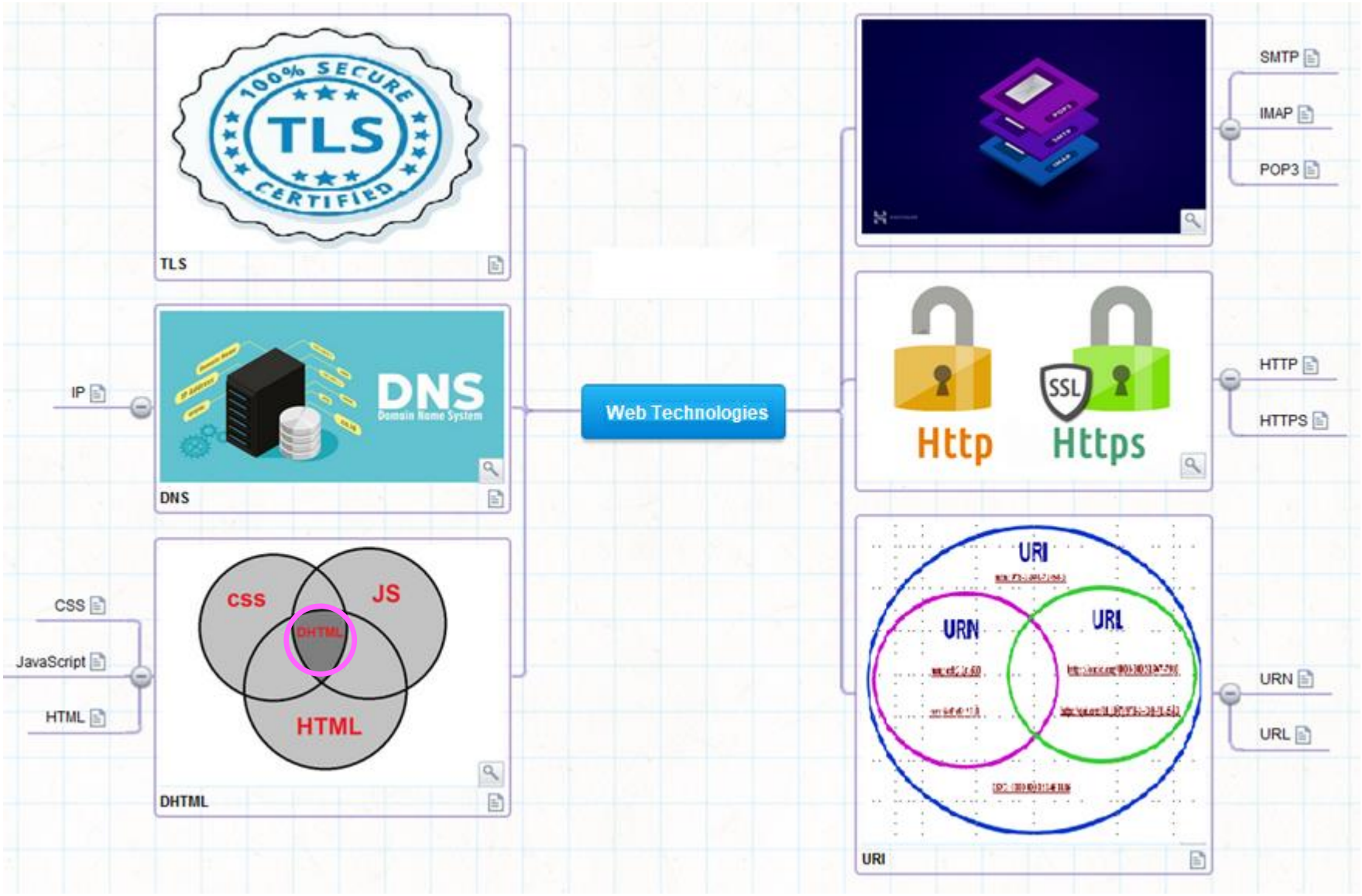
- Links sobre HTML :
  - https://html.spec.whatwg.org
  - http://www.w3schools.com/html

DHTML: Descrição Breve

- DHTML stands for Dynamic HTML.
- "Dynamic" is defined as the ability of the browser alter a web page's look and style after the document has loaded.
- It just uses languages features to build dynamic web pages.

W3C once said: *"Dynamic HTML is a term used by some vendors to describe the combination of HTML, style sheets and scripts that allows documents to be animated."*

DHTML = HTML+JS+CSS



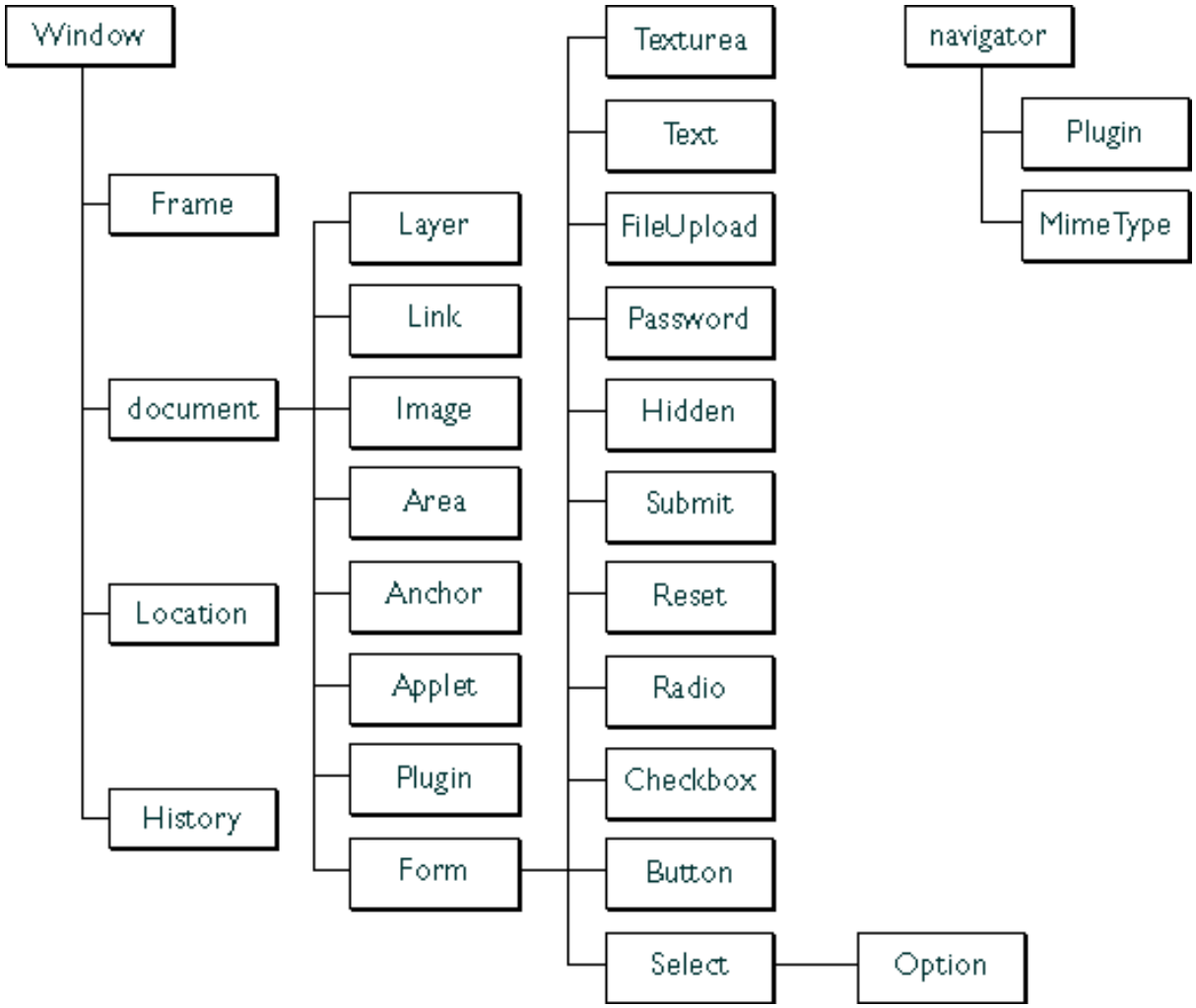
DHTML: Características

- Simplest feature is making the page dynamic.
- Can be used to create animations, games, applications.
- Dynamic building of web pages is simple as no plug-in is required.
- Facilitates the usage of events, methods and properties and code reuse.
- It makes the Web experience faster and more interactive for end users.

DOM: Modelo de Objetos do Documento

- All these three components are linked via Document Object Model (DOM).
- Document Object Model is to provide a standard programming interface.
- DHTML is NOT a scripting language
- It uses different technologies.

DOM: Árvore do Documento HTML



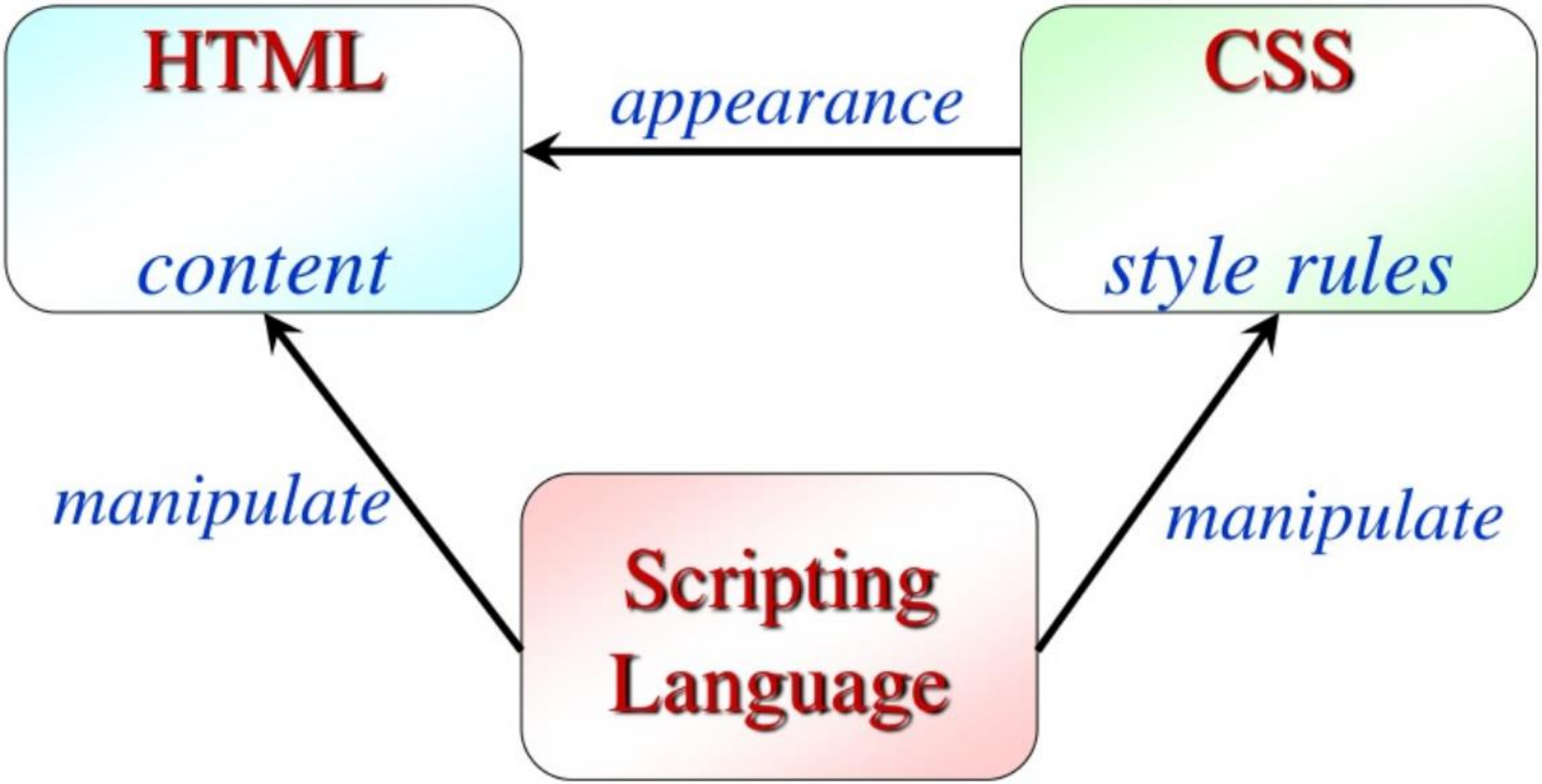
checkValidity() - Returns true if an input element contains valid data

setCustomValidity() - Sets the validationMessage property of an input element

DOM: Vantagens

- The Document Object Model is a platform- and language-neutral interface.
- Defines a hierarchical model of the document structure through which all document elements may be accessed.
- Relatively simple to modify data structure and extract data.

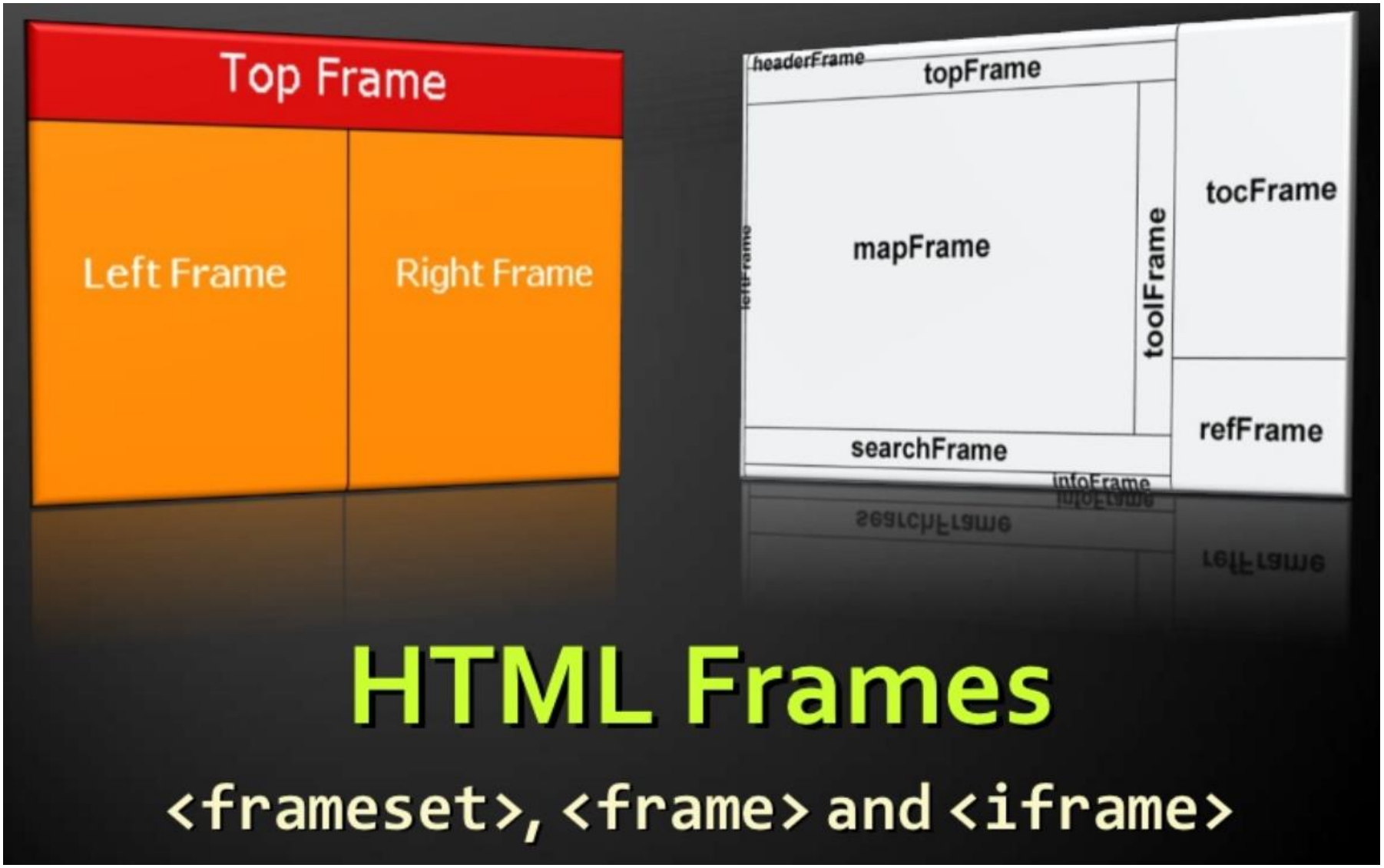
DHTML: Papel dos Componentes Tecnológicos



DHTML: Vantagens

- DHTML can make your browser dynamic and interactive.
- Validation of input's given by the user can be done at the client side, without connection to the server.
- Content and design can be separated using Style sheets & uniformity of the site can be maintained using them.

HTML – Frames


Top Frame
Left Frame
Right Frame
headerFrame
topFrame
mapFrame
toolFrame
tocFrame
refFrame
searchFrame
infoFrame
HTML Frames
<frameset>, <frame> and <iframe>

HTML – *Frames*

- Frames provide a way to show multiple HTML documents in a single Web page
- The page can be split into separate views (frames) horizontally and vertically
- Frames were popular in the early ages of HTML development, but now their usage is rejected
- Frames are not supported by all user agents (browsers, search engines, etc.)
  - A <noframes> element is used to provide content for non-compatible agents.

HTML – Frameset e iFrame

Deprecated HTML5

```
<html>
  <head><title>Frames Example</title></head>

  <frameset cols="180px,*,150px">
    <frame src="left.html" />
    <frame src="middle.html" />
    <frame src="right.html" />
  </frameset>

</html>
```

- Inline frames provide a way to show one website inside another website:

```
<iframe name="iframeGoogle" width="600" height="400"
src="http://www.google.com" frameborder="yes"
scrolling="yes"></iframe>
```

HTML – Exemplo de *Frameset*

- A página tem definidas duas molduras verticais:

[testeframe.html]

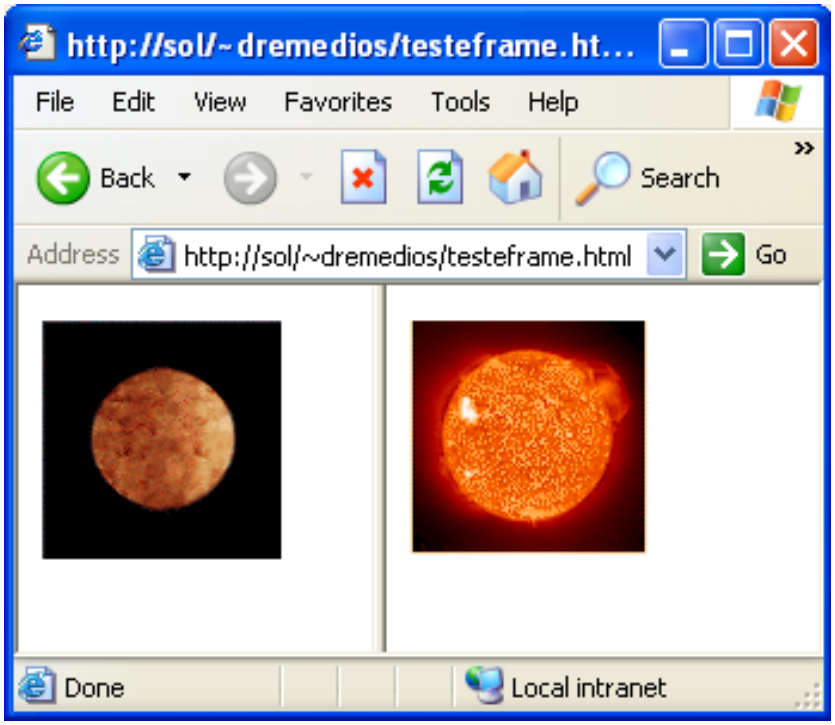
```
<html>
  <head>
  </head>
  <frameset cols = “150,*”>
    <frame src ="mercur.html" noresize = "noresize">
    <frame src ="sun.html" noresize = "noresize">
    <noframes>Sorry! Your browser does not support frames</noframes>
  </frameset>
</html>
```

[mercur.html]

```
<HTML>
<HEAD></HEAD>
<BODY>
<P><IMG height=“100” src="img/merglobe.gif" width=“100”
border=“0”>
</P>
</BODY>
</HTML>
```

[sun.html]

```
<HTML>
<HEAD></HEAD>
<BODY>
<P><IMG height=“98” src="img/sun.gif" width=“99” border=“0”>
</P>
</BODY>
</HTML>
```



Não suportado em HTML5

HTML: Formulários

- HTML Forms are required when you want to collect some data from the site visitor
- For example during user registration you would like to collect information such as name, email address, credit card, etc.
- A form will take input from the site visitor and then will post it to a back-end application
- The back-end application will perform required processing on the passed data based on defined business logic inside the application

HTML: Formulário

- **<form>** … **</form>** – Delimita um formulário
  - **Action** – Indica o *URL para submeter os dados (Servlet ou JSP)*
  - **Method** – Especifica o método de envio dos dados
    - Method = "post" – recomendado, silencioso, maior dimensão
    - Method = "get" – usar só em desenvolvimento e testes (URL)
  - **Target** – Assinala a *frame* onde será visualizada a resposta
    - Pode ser uma *frame* especifica referindo o seu nome
      - Target = "my_frame"
    - Pode ser uma *frame* relativa
      - Target = "_blank" – Nova janela
      - Target = "_self " – Na própria *frame*
      - Target = "_parent" – No *frameset* pai
      - Target = "_top" – No corpo da própria janela

Formulário: GET ou POST

```
<form action="http://www.foo.com" method="GET | POST">
  <div>
    <label for="say">What greeting do you want to say?</label>
    <input name="say" id="say" value="Hi">
  </div>
  <div>
    <label for="to">Who do you want to say it to?</label>
    <input name="to" id="to" value="Mom">
  </div>
  <div>
    <button>Send my greetings</button>
  </div>
</form>
```

```
GET /?say=Hi&to=Mom HTTP/2.0
Host: foo.com
```

Sending form data

```
POST / HTTP/2.0
Host: foo.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 13

say=Hi&to=Mom
```

As forms are the main tool for data input in Web applications, and the data we want to collect has become more complex, it has been necessary to create an input element with more capabilities, to collect this data with more semantics and better definition, and allow for easier, more effective error management and validation.

HTML: Elementos do Formulário

- There are different types of form controls that you can use to collect data using HTML form:
  - Text Input Controls
  - Checkboxes Controls
  - Radio Box Controls
  - Select Box Controls
  - File Select boxes
  - Hidden ControlsClickable Buttons
  - Submit and Reset Button

HTML: Elemento Select Box

- **A select box, also called drop down box which provides option to list down various options in the form of drop down list, from where a user can select one or more options.**

```
<html>
<head>
<title>Select Box Control</title>
</head>
<body>
<form>
<select name="dropdown">
<option value="Maths" selected>Maths</option>
<option value="Physics">Physics</option>
</select>
</form>
</body>
</html>
```

HTML: Elemento Select Box

- **A select box, also called drop down box which provides option to list down various options in the form of drop down list, from where a user can select one or more options.**

```
<html>
<head>
<title>Select Box Control</title>
</head>
<body>
<form>
<select name="dropdown">
<option value="Maths" selected>Maths</option>
<option value="Physics">Physics</option>
</select>
</form>
</body>
</html>
```

HTML: Atributos Select Box

Attribute	Description
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
size	This can be used to present a scrolling list box.
multiple	If set to "multiple" then allows a user to select multiple items from the menu.

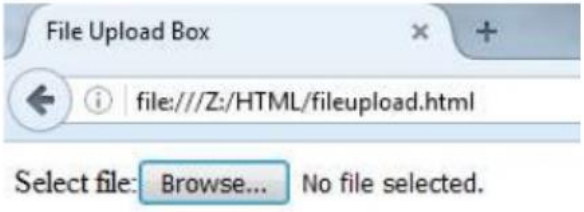
Following is the list of important attributes of <option> tag:

Attribute	Description
value	The value that will be used if an option in the select box box is selected.
selected	Specifies that this option should be the initially selected value when the page loads.
label	An alternative way of labeling options

HTML: Elemento File Upload Box

- If you want to allow a user to upload a file to your web site, you will need to use a file upload box, also known as a file select box
- This is created using the <input> element but type attribute is set to file

```
<html>
<head>
<title>File Upload Box</title>
</head>
<body><form>
Select file:
<input type="file" name="fileupload" accept="image/*" />
</form>
</body>
</html>
```



HTML: Botões

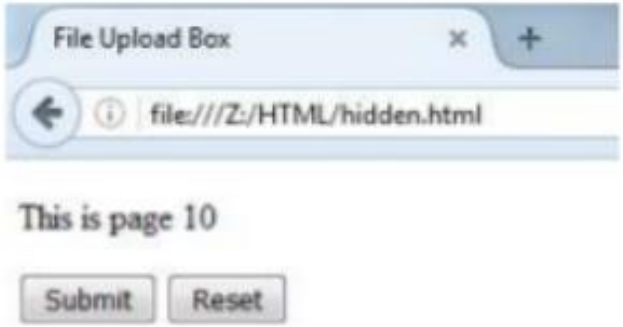
- There are various ways in HTML to create clickable buttons

```
<html>
<head>
<title>Buttons</title>
</head>
<body>
<form>
<input type="submit" name="submit" value="Submit" />
<input type="reset" name="reset"  value="Reset" />
<input type="button" name="ok" value="OK"  />
<input type="image" name="imagebutton" src="images/logo.jpg" />
</form>
</body>
</html>
```

HTML: Elementos Escondidos

- Hidden form controls are used to hide data inside the page which later on can be pushed to the server

```
<html>
<head>
<title>File Upload Box</title>
</head>
<body>
<form>
<p>This is page 10</p>
<input type="hidden" name="pagename" value="10" />
<input type="submit" name="submit" value="Submit" />
<input type="reset" name="reset"  value="Reset" />
</form>
</body>
</html>
```



Exemplo de Validação de Formulários

Alias:
3 checks: Not blank, at least 3 characters, only alphanumeric characters allowed.

Password: Verify Password:
2 checks: Require at least 5 characters in the first password field. Check if both password field values are the same.

Comment: (Please enter 150 characters maximum)

Gender: Select a Gender
2 checks: Check that at least one drop down has been selected. The first drop down is an invalid selection.

Email:
2 checks: Can't be blank. A valid email address must contain an "@" and a "." in the address.

Numbers:
2 checks: Can't be blank. Only digits 0-9 accepted.

Fruit:
- A. Apples
- B. Oranges
- C. Pears
- D. Lemons

1 check: At least one radio button must be selected.

Check Box:
1 check: Reminds the user they have not checked the box. Does not fail the validation however.

Submit Reset

Exemplo de Validação de Formulários (1)

```
<html>
<head>
<script>

function Validator(theForm)
{
/*ALIAS*/

  // check to see if the field is blank
  if (theForm.Alias.value == "")
  { alert("You must enter an
      alias.");
    theForm.Alias.focus();
    return false;
  }

  // require at least 3 characters be
      entered
  if (theForm.Alias.value.length < 3)
  {alert("Please enter at least 3
      characters in the \"Alias\"
      field.");
    theForm.Alias.focus();
    return false;
  }
```

```
// allow ONLY alphanumeric keys, no symbols or punctuation
// this can be altered for any "checkOK" string you desire
var checkOK =
    "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz01234
    56789";
var checkStr = theForm.Alias.value;
var allValid = true;
for (i = 0;  i < checkStr.length;  i++)
{ch = checkStr.charAt(i);
  for (j = 0;  j < checkOK.length;  j++)
  if (ch == checkOK.charAt(j))
  break;
  if (j == checkOK.length)
 {allValid = false;
   break;
  }
}
if (!allValid)
{alert("Please enter only letter and numeric characters in the  \"Alias\" field.");
  theForm.Alias.focus();
  return (false);
}
```

Exemplo de Validação de Formulários (2)

```
/*PASSWORD*/
  // require at least 5 characters in the password field
  if (theForm.Password.value.length < 5)
  {alert("Please enter at least 5 characters in the \"Password\" field.");
    theForm.Password.focus();
    return false;
  }

  // check if both password fields are the same
  if (theForm.Password.value != theForm.Password2.value)
  {alert("The two passwords are not the same.");
    theForm.Password2.focus();
    return false;
  }

/*COMMENT*/
  // allow only 150 characters maximum in the comment field
  if (theForm.comment.value.length > 150)
  {alert("Please enter at most 150 characters in the comment field.");
    theForm.comment.focus();
    return false;
  }

/*GENDER*/
  // check if no drop down has been selected
  if (theForm.sex.selectedIndex < 0)
  {alert("Please select one of the \"Gender\" options.");
    theForm.sex.focus();
    return false;
  }

  // check if the first drop down is selected, if so, invalid selection
  if (theForm.sex.selectedIndex == 0)
  {alert("The first \"Gender\" option is not a valid selection.");
    theForm.sex.focus();
    return false;
  }

/*EMAIL*/
  // check if email field is blank
  if (theForm.Email.value == "")
  {alert("Please enter a value for the \"Email\" field.");
    theForm.Email.focus();
    return false;
  }
  // test if valid email address, must have @ and .
  var checkEmail = "@.";
  var checkStr = theForm.Email.value;
  var EmailValid = false;
  var EmailAt = false;
  var EmailPeriod = false;
```

Exemplo de Validação de Formulários (3)

```
for (i = 0;  i < checkStr.length;  i++)
 {ch = checkStr.charAt(i);
  for (j = 0;  j < checkEmail.length;  j++)
  {if (ch == checkEmail.charAt(j) && ch == "@")
    EmailAt = true;
   if (ch == checkEmail.charAt(j) && ch == ".")
     EmailPeriod = true;
   if (EmailAt && EmailPeriod)
     break;
   if (j == checkEmail.length)
     break;
  }
  // if both the @ and . were in the string
  if (EmailAt && EmailPeriod)
  {EmailValid = true
    break;
  }
 }
 if (!EmailValid)
 {alert("The \"email\" field must contain an \"@\" and a \".\".");
  theForm.Email.focus();
  return false;
 }
```

```
/*NUMBERS*/

 // check if numbers field is blank
 if (theForm.numbers.value == "")
 {
  alert("Please enter a value for the \"numbers\" field.");
  theForm.numbers.focus();
  return false;
 }

// only allow numbers to be entered
 var checkOK = "0123456789";
 var checkStr = theForm.numbers.value;
 var allValid = true;
 var allNum = "";
 for (i = 0;  i < checkStr.length;  i++)
 {
  ch = checkStr.charAt(i);
  for (j = 0;  j < checkOK.length;  j++)
    if (ch == checkOK.charAt(j))
      break;
  if (j == checkOK.length)
  {
    allValid = false;
    break;
  }
  if (ch != ",")
    allNum += ch;
 }
```

Exemplo de Validação de Formulários (4)

```
 if (!allValid)
  {alert("Please enter only digit characters in the \"numbers\" field.");
    theForm.numbers.focus();
    return (false);
  }
// require at least one radio button be selected
  var radioSelected = false;
  for (i = 0;  i < theForm.fruit.length;  i++)
  {if (theForm.fruit[i].checked)
    radioSelected = true;
  }
  if (!radioSelected)
  { alert("Please select one of the \"Fruit\" options.");
    return (false);
  }

/*CHECKBOX*/
  // alert if the box is NOT checked
  if (!theForm.checkbox1.checked)
    alert("Just reminding you that if you wish to have our Super Duper option,
          \nyou must check the box!");
  alert("All Validations have succeeded. ");
  return true;   // submission
  }
</script>
</head>
```

```
<body>
<form action = “hello.html"
method=“GET" onsubmit="return Validator(this)" name="Form1">
Alias:
      <input type="text" size="15" maxlength="15"
      name="Alias"><br>
      3 checks: Not blank, at least 3 characters, only alphanumeric
      characters allowed.<br><br>

Password:
      <input type="password" size="10" maxlength="10"
      name="Password">

Verify Password:
      <input type="password" size="10" maxlength="10"
      id="Password2"><br>
      2 checks: Require at least 5 characters in the first password field.
      Check if both password field values are the same.<br><br>

Comment: (Please enter 150 characters maximum)<br>
<textarea name="comment" rows="4" cols="50" wrap="virtual">
</textarea><br><br>
```

Exemplo de Validação de Formulários (5)

```
Gender:
        <select name="sex" size="1">
        <option>Select a Gender</option>
        <option value="M">Male</option>
        <option value="F">Female</option>
        </select><br>
2 checks: Check that at least one drop down has been selected. The first drop down is an invalid selection.<br><br>

Email:
        <input type="text" size="60" name="Email" value= ="" ><br>
        2 checks: Can't be blank. A valid email address must contain an "@"  and a "." in the address.<br><br>

Numbers:
        <input type="text" size="3" maxlength="3" name="numbers"><br>
        2 checks: Can't be blank. Only numbers 0-9 accepted.<br><br>

Fruit:<br>
        <input type="radio" name="fruit" value="A"> A. Apples<br>
        <input type="radio" name="fruit" value="B"> B. Oranges<br>
        <input type="radio" name="fruit" value="C"> C. Pears<br>
        <input type="radio" name="fruit" value="D"> D. Lemons<br>
        1 check: At least one radio button must be selected.<br><br>

Check Box: <input type="checkbox" name="checkbox1" value="Y"><br>
        1 check: Reminds the user they have not checked the box. Does not fail the validation however.<br><br>
        <input type="submit" name="Submit" value="Submit">
        <input type="reset" name="Reset" value="Reset"><br>
</form>
</body></html>
```

NEW FORM ELEMENTS,
ATTRIBUTES & TYPES

Introdução

- HTML5 web forms introduced new form elements, input types, attributes ,and other features.
- Many features using in our interfaces: form validation, combo boxes, placeholder text, and the like.
- Marking up forms easier on the developer, also better for the user.

HTML5: Formulário

The **HTML** `<form>` **element** represents a document section containing interactive controls for submitting information.

```
<form action="" method="get" class="form-example">
  <div class="form-example">
    <label for="name">Enter your name: </label>
    <input type="text" name="name" id="name" required>
  </div>
  <div class="form-example">
    <label for="email">Enter your email: </label>
    <input type="email" name="email" id="email" required>
  </div>
  <div class="form-example">
    <input type="submit" value="Subscribe!">
  </div>
</form>
```

HTML: Elementos do Formulário

Form-associated elements mean that the element can have a *form owner*. The form-associated elements can be:

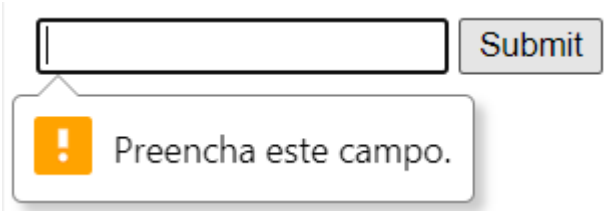
- **<button> creates a button control in an HTML document**
- **<fieldset> used for grouping elements in an HTML document**
- **<input> form control to allow the user to edit the data**
- **<label> caption for a form control**
- **<object> external resource embedded in an HTML document**
- **<select> control for selecting a set of options**
- **<textarea> used to allow multiple lines of text**
- **<img> image embedded in an HTML document**

A form can contain other elements too — it doesn't have to be a form-associated element in order to be nested. For example, you can include <p>, and elements within a form.

Validação: Obrigatoriedade

- A validação de formulários é muito importante tanto no lado do cliente como no lado do servidor
- Ajuda os utilizadores legítimos a evitar e corrigir erros e impede que os utilizadores maliciosos enviem dados inconsistentes
- A validação assenta na indicação do formato dos valores esperados nos vários controlos dos formulários
  - Para além do tipo, por exemplo, limites superiores/inferiores
- Para que os controlos de formulário sejam validados, precisam de ter um atributo nome, pois sem o qual não serão submetidos
- Uma validação comum é a obrigatoriedade de preenchimento, não permite que um formulário seja submetido com dados por preencher

```
<input name="myInput" required>
```



Validação *Client-Side* (*type&pattern*)

- A validação de dados no lado do cliente pode ser realizada especificando tipos (*type*) pré-definidos em vez de simplesmente definir entradas de texto
- Podem criar-se explicitamente entradas para dados, tais com:
  - números, endereços de e-mail ou URLs.
- Os navegadores validam automaticamente os dados inseridos pelo utilizador
- Os navegadores usam um padrão incorporado que define o critério de validação notificando o utilizador se os dados não correspondem

- Para dados com uma determinada estrutura (por exemplo, formulários com identificadores de utilizadores que só podem conter uma sequência específica de letras minúsculas e números), pode ser usado o atributo padrão (*pattern*) para especificar uma expressão regular personalizada

```
<input type="text" … pattern="[a-z]{3}[0-9]{3}">
```

Formulário: Tipos de <input>

HTML5 gives us input types that provide for more data-specific UI elements and native data validation.

Type	Data Type
Search	Text with no line breaks
Telephone	Text with no line breaks
URL	An absolute URL
E-mail	An e-mail address or list of e-mail addresses
Date	A date (year, month, day) with no time zone
Month	A date consisting of a year and a month with no time zone
Week	A date consisting of a week-year number and a week number with no time zone
Time	A time (hour, minute, seconds, fractional seconds) with no time zone
Local Date and Time	A date and time (year, month, day, hour, minute, second, fraction of second) with no timezone offset
Number	A numerical value
Range	A numerical value, with the extra semantic that the exact value is not important
Color	An RGB color with 8-bit red, green, and blue components

Tipo: <input type="number&range">

<input type="number">

The first new input type we'll discuss is type="number":

This creates a special kind of input field for number entry – in most supporting browsers this appears as a text entry field with a spinner control, which allows you to increment and decrement its value.

<input type="range">

Creating a slider control to allow you to choose between a range of values used to be a complicated, semantically dubious proposition, but with HTML5 it is easy — you just use the rangeinput type:

Tipo: <input type="date&time">

<input type="date"> and other date/time controls

HTML5 has a number of different input types for creating complicated date/time pickers, for example the kind of date picker you see featured on pretty much every flight/train booking site out there. These used to be created using unsemantic kludges, so it is great that we now have standardized easy ways to do this.

Respectively, these create a fully functioning date picker, and a text input containing a separator for hours, minutes and seconds (depending on the step attribute specified) that only allows you to input a time value.

date and time input types.

```
<input type="date" … >
<input type="time" … >
```

Tipo: <input type="color">

<input type="color">
This input type brings up a color picker. Opera's implementation allows the user to pick from a
selection of colors, enter hexadecimal values directly in a text field, or to invoke the OS's native color picker.
a color input, and the native color pickers on Windows and OS X.

Tipo: <input type="tel&email&url">

- <input type="tel"> <input type="email"> e <input type="url">
- Tal como os seus nomes sugerem, estes tipos de entrada referem números de telefone, endereços de e-mail e URLs
- O navegador vai considerá-los como texto
- No entanto, indicar o tipo que se espera desempenha um papel importante na validação do formato
- Além disso, em certos dispositivos móveis, o navegador mudará do teclado de entrada de texto regular para variantes mais relevantes para o contexto
- No futuro, provavelmente, os navegadores irão aproveitar estes tipos para oferecerem funcionalidades adicionais, como por exemplo, completar automaticamente endereços de e-mail ou números de telefone usando a lista de contactos do utilizador

Novos Atributos: form & input

- New attributes for <form>:
  - autocomplete
  - novalidate

HTML Form Tag

- New attributes for <input>:
  - autocomplete
  - autofocus
  - form
  - formaction
  - formenctype
  - formmethod
  - formnovalidate
  - formtarget
  - height and width
  - list
  - min and max
  - multiple
  - pattern (regexp)
  - placeholder
  - required
  - step

HTML Input Tag

Além de tipos explícitos de entrada,
o HTML5 define atributos para controlar formulários
que ajudam a simplificar algumas tarefas comuns,
tais como, especificar os valores esperados
para determinados dados

Novos Atributos: <input autofocus&min/max>

Autofocus

Another common feature that previously had to rely on scripting is having a form field automatically focused when a page is loaded. This can now be achieved with the autofocusattribute:

```
<input type="text" autofocus … >
```

Keep in mind that you shouldn't have more than one autofocus form control on a single page. You should also use this sort of functionality with caution, in situations where a form represents the main area of interest in a page. A search page is a good example – provided that there isn't a lot of content and explanatory text, it makes sense to set the focus automatically to the text input of the search form.

Min and Max

As their name suggests, this pair of attributes allows you to set a lower and upper bound for the values that can be entered into a numerical form field, for example number, range, time or date input types (yes, you can even use it to set upper and lower bounds for dates – for instance, on a travel booking form you could limit the datepicker to only allow the user to select future dates). For range inputs, min and max are actually necessary to define what values are returned when the form is submitted.

```
<input type="number" … min="1" max="10">
```

Novos Atributos: <input placeholder&step>

Placeholder

A common usability trick in web forms is to have placeholder content in text entry fields – for instance, to give more information about the expected type of information we want the user to enter – which disappears when the form control gets focus. While this used to require some JavaScript (clearing the contents of the form field on focus and resetting it to the default text if the user left the field without entering anything), we can now simply use the placeholder attribute:

```
<input type="text"… placeholder="John Doe">
```

Step

The step attribute can be used with a numerical input value to dictate how granular the values you can input are. For example, you might want users to enter a particular time, but only in 30 minute increments. In this case, we can use the step attribute, keeping in mind that for time inputs the value of the attribute is in seconds:

```
<input type="time" … step="1800">
```

Novos Elementos do Formulário

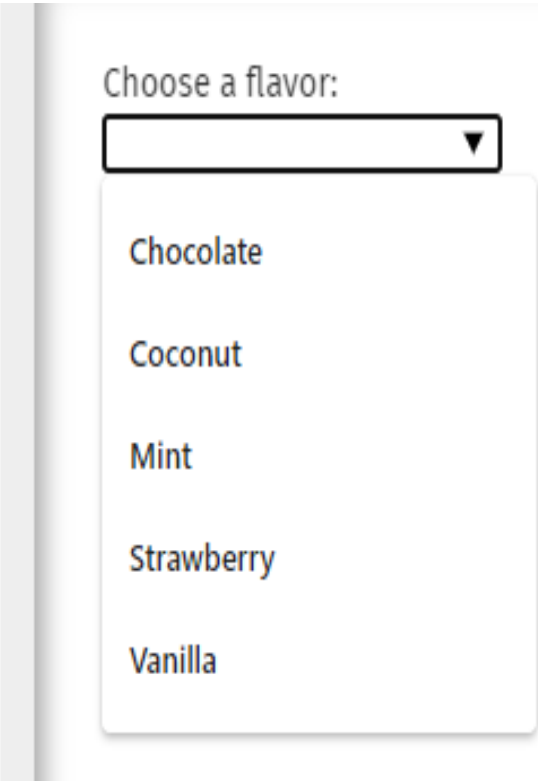
HTML5 added the following form elements:

1. <datalist>
2. <keygen> **(Deprecated)**
3. <output>

Formulário: *Datalist*

- The **<datalist>** element specifies a list of pre-defined options for an <input> element.
- The list is created with <option> elements inside the <datalist>

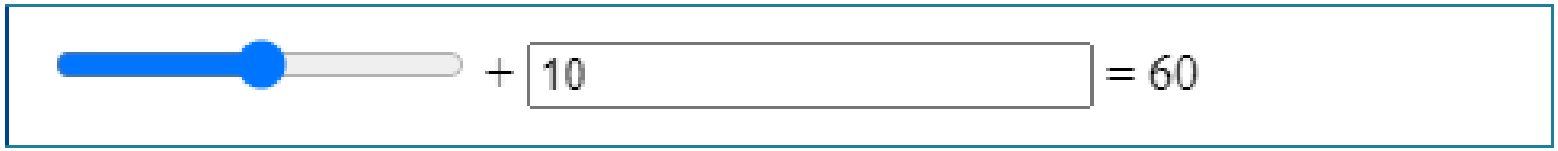
```
<label for="ice-cream-choice">Choose a flavor:</label>
<input list="ice-cream-flavors" id="ice-cream-choice" name="ice-
cream-choice" />

<datalist id="ice-cream-flavors">
    <option value="Chocolate">
    <option value="Coconut">
    <option value="Mint">
    <option value="Strawberry">
    <option value="Vanilla">
</datalist>
```



Formulário: Output

The <output> tag represents the result of a calculation

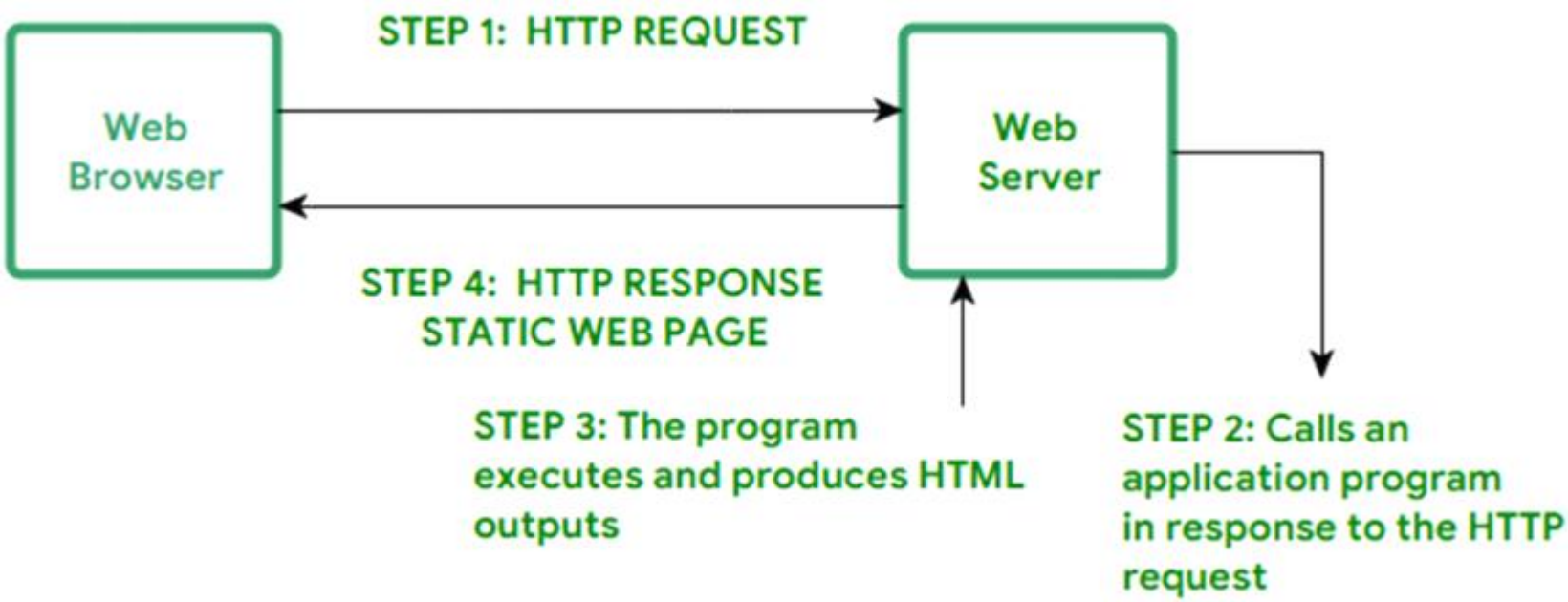
```
<form oninput="result.value=parseInt(a.value)+parseInt(b.value)">
  <input type="range" id="b" name="b" value="50" /> +
  <input type="number" id="a" name="a" value="10" /> =
  <output name="result" for="a b">60</output>
</form>
```



HTML5: Dúvida Metódica

- *Resta saber se é a altura certa para incluir uma API que 'ainda' não é completamente suportada pelos navegadores…*
- *Por outro lado, existem muitas coisas que não são completamente suportadas e são usadas…*
- *Provavelmente, é preciso considerar a versão do navegador e agir em conformidade…*
- *Inevitavelmente, é preciso validar os dados do lado do servidor!*

Website: Estático vs. Dinâmico

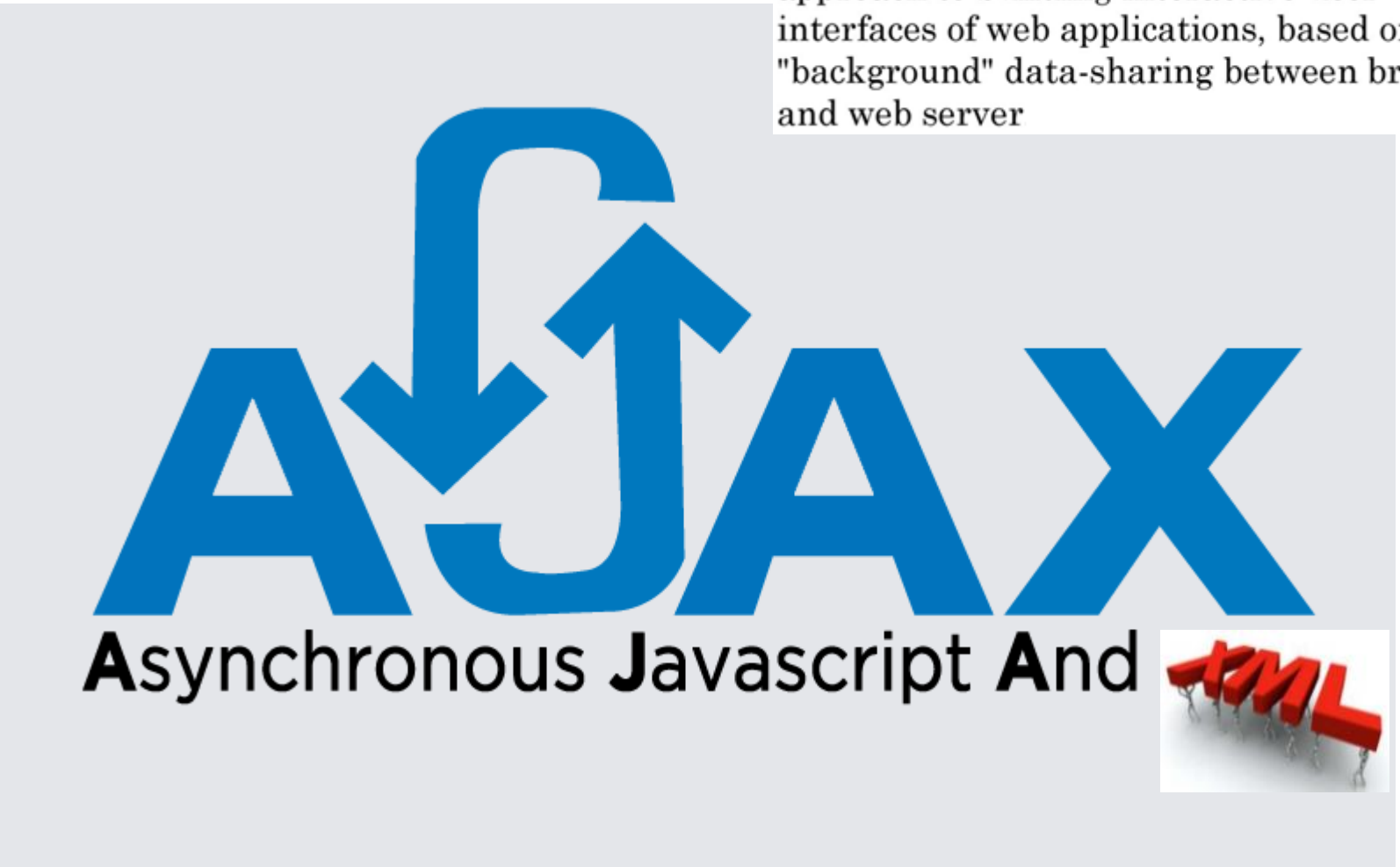


Dynamic web pages

- Html page is generated dynamically
- Interaction with user
- Becomes slower as functionality increases
- Speed becomes intolerable, so AJAX has been born

AJAX

approach to building interactive user interfaces of web applications, based on the "background" data-sharing between browser and web server

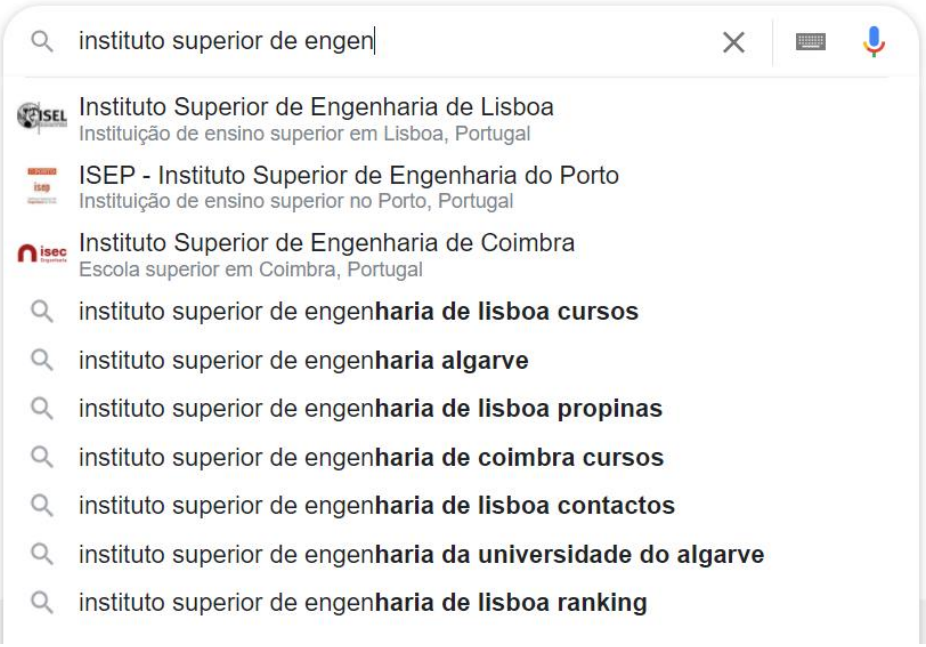


AJAX: Google Suggest

The term AJAX is coined on February 18, 2005, by **Jesse James Garret** in a short essay published a few days after Google released its Maps application.

In the year 2006, the W3C (World Wide Web Consortium) announces the release of the first draft which made AJAX an official web standard.

Google Suggest is using AJAX to create a very dynamic web interface: When you start typing in Google's search box, a JavaScript sends the letters off to a server and the server returns a list of suggestions.



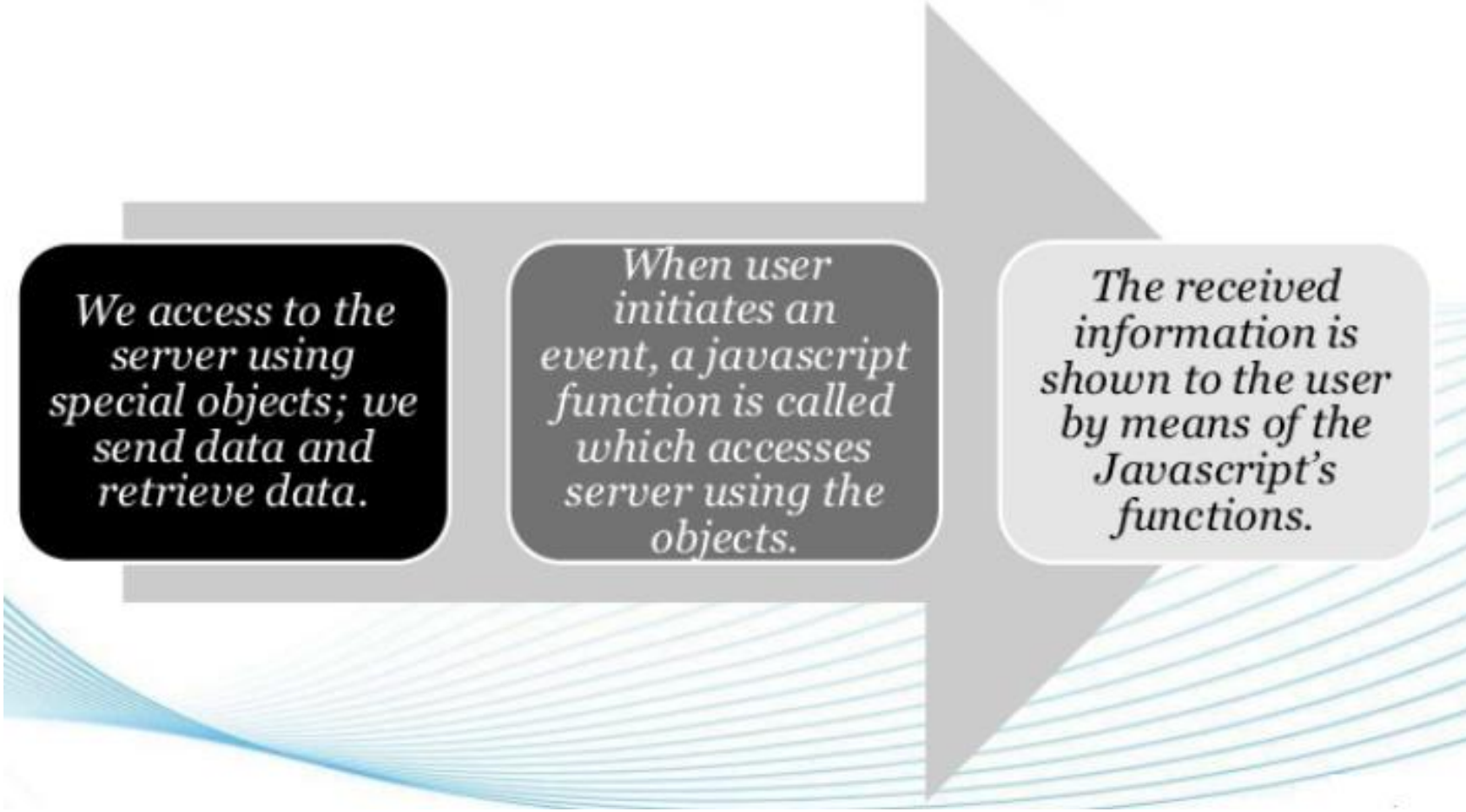
SO WHAT IS AJAX ???

- A programming language – no...
- A new technology – not exactly...
- So what else ?

It is a methodology on using several web technologies together, in an effort to close the gap between the usability and interactivity of a desktop application and the ever demanding web application

AJAX: JavaScript

- AJAX is based on Javascript, and the main functionality is to access the web server inside the Javascript code.



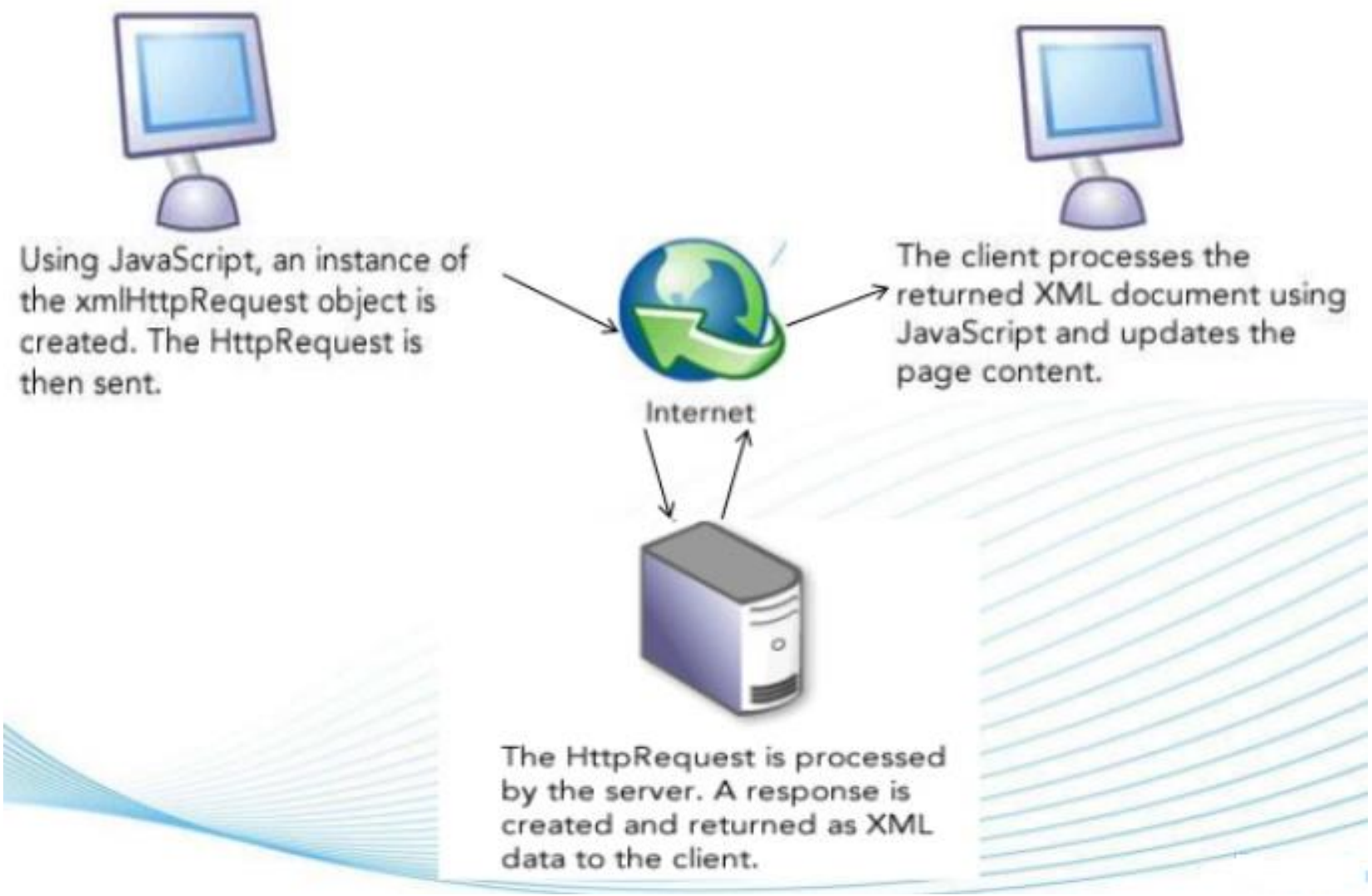
AJAX: Assíncrono (síncrono)

- Asynchronous Javascript and XML is a **client side** techniques that combines a set of known technologies in order to create faster and more user friendly web pages.
- AJAX provides an ability to communicate with the server asynchronously.

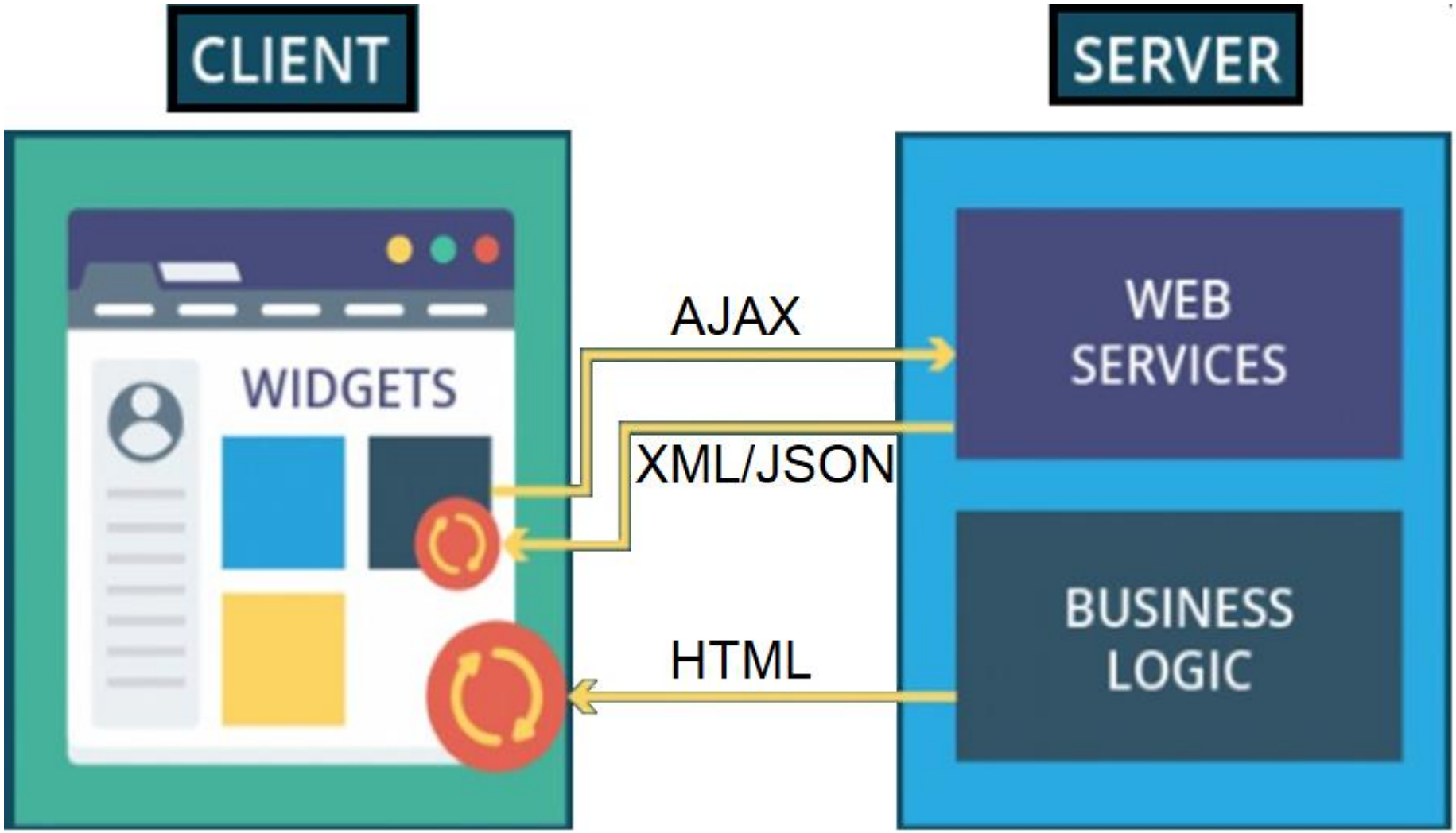
ASYNCHRONOUS???

We can send a request to server and continue user interaction with the user with out waiting for server response. Once the response arrives, a designated area in UI will update itself and reflect the response information. Whole page need not be reloaded

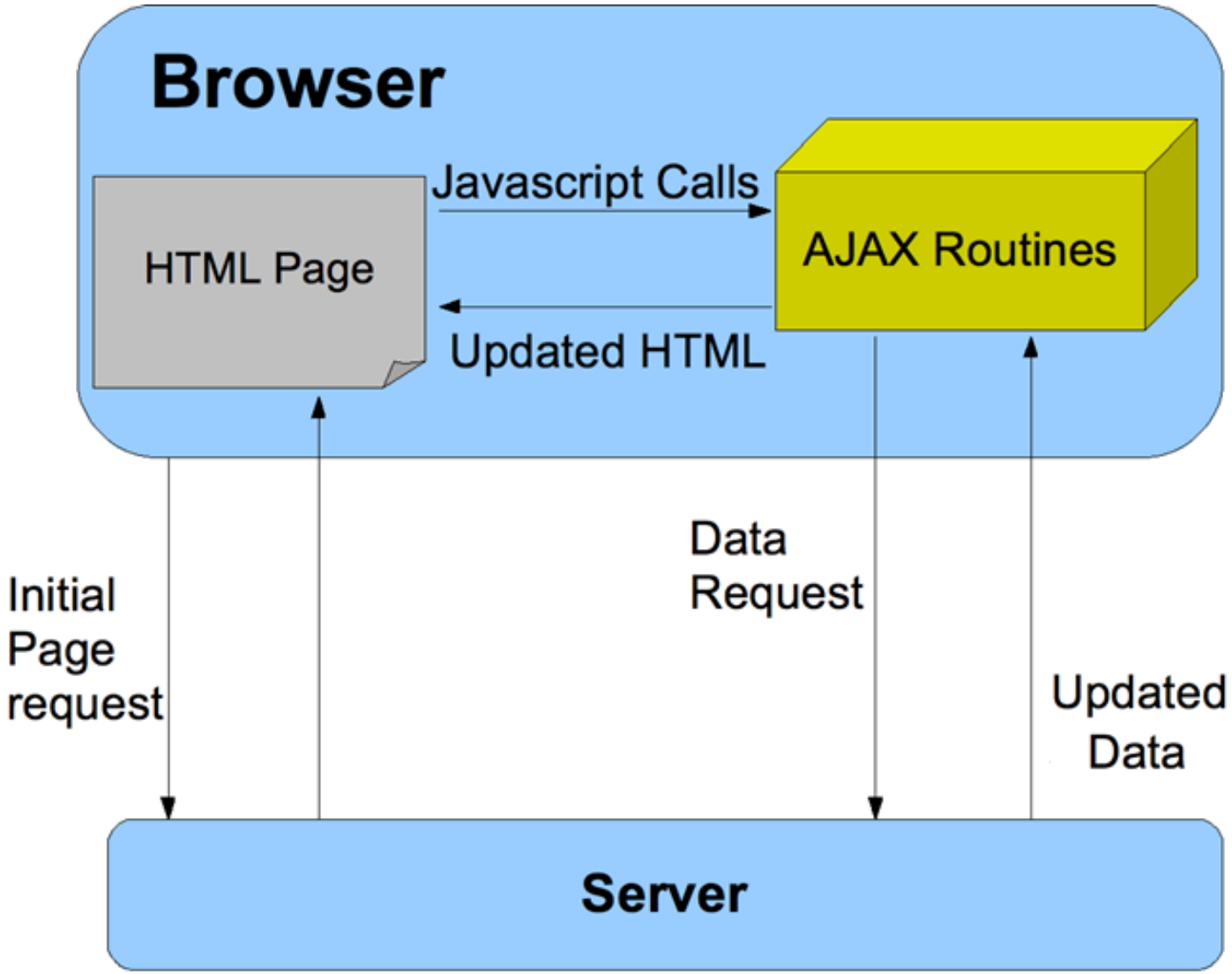
AJAX: Ciclo de Processamento



AJAX: Interoperabilidade

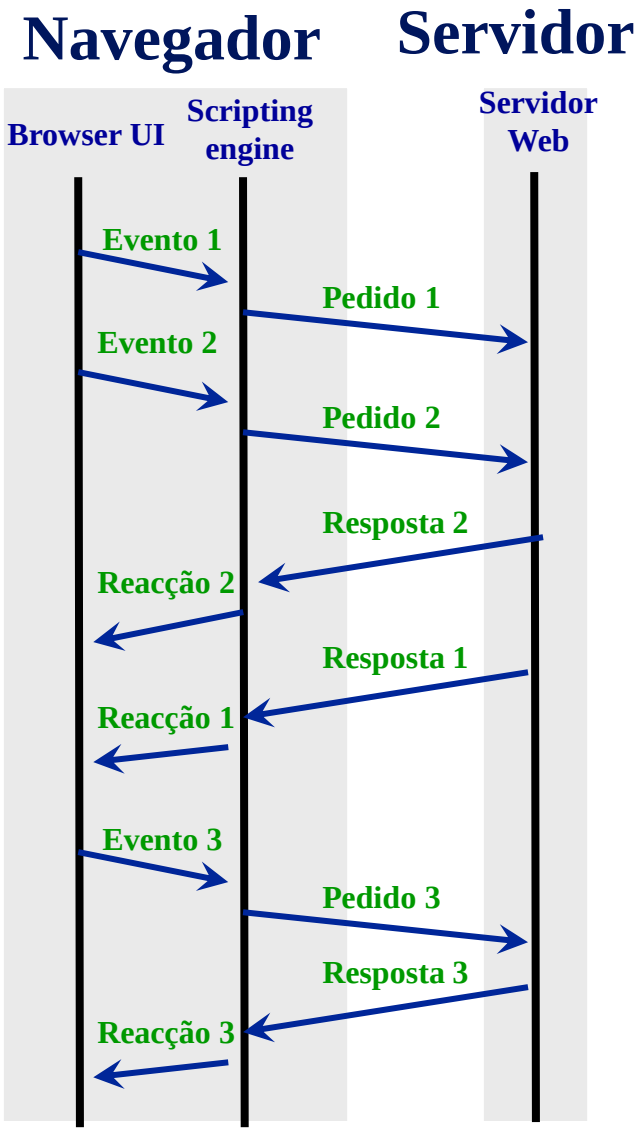


AJAX: Interação com Servidor



AJAX: Asynchronous JavaScript And XML

- A experiência de utilização das aplicações é melhorada, mais *user-friendly*, porque não se perde o contexto de utilização da página
- Permite atualizar parcialmente uma página sem forçar o seu carregamento total, só transfere o que é necessário
- Realiza transferência assíncrona (ou síncrona) de dados entre o *browser* e o servidor usando o objeto XMLHttpRequest
- O servidor não sabe que está a responder a um pedido HTTP via AJAX



AJAX: XMLHttpRequest Object

- The kernel of Ajax ⟶ XmlHttp Request Object

- The XMLHttpRequest object allows client-side *JavaScript* to make HTTP requests (both GET and POST) to the server without reloading pages in the browser.

- This JavaScript object was originally introduced in Internet Explorer 5 by Microsoft and it is the enabling technology that allows asynchronous requests

AJAX: *XMLHttpRequest*

Important Methods

Open("method","url",boolean)

- to send a request to server
- ***method*** - GET or POST
- ***url*** - address of the target file
- ***boolean*** - to denote whether the request is synchronous or asynchronous. If true, asynchronous.

Send(content)

- to send data to server for processing
- ***content*** - may be string or DOM type of document.

AJAX: *XMLHttpRequest*

Important Properties

- ***readystate***

  - used to identify the state of the request. Possible values 0-4.

- ***onreadystatechange***

  - event handler for an event that fires at every state change.

- ***status***

  -Numeric code return by server.Eg.404,200 **(Status HTTP)**

- ***responseText***

  - the string data returned by the server process.

- ***responseXML***

  - the DOM type of document returned by the server process.

AJAX: Estado do Pedido

- Objeto **XMLHttpRequest (XHR)**
  - Propriedade **onreadystatechange –** Permite registar a função que será invocada em todas as mudanças de estado do pedido (tipicamente só interessa o 'complete')

```
xmlHttp.onreadystatechange=function() {
   // tratar a resposta
}
```

  - Propriedade **readyState –** indica o estado do pedido

State	Description
0	The request is not initialized
1	The request has been set up
2	The request has been sent
3	The request is in process
4	**The request is complete**

AJAX: Pedido e Resposta

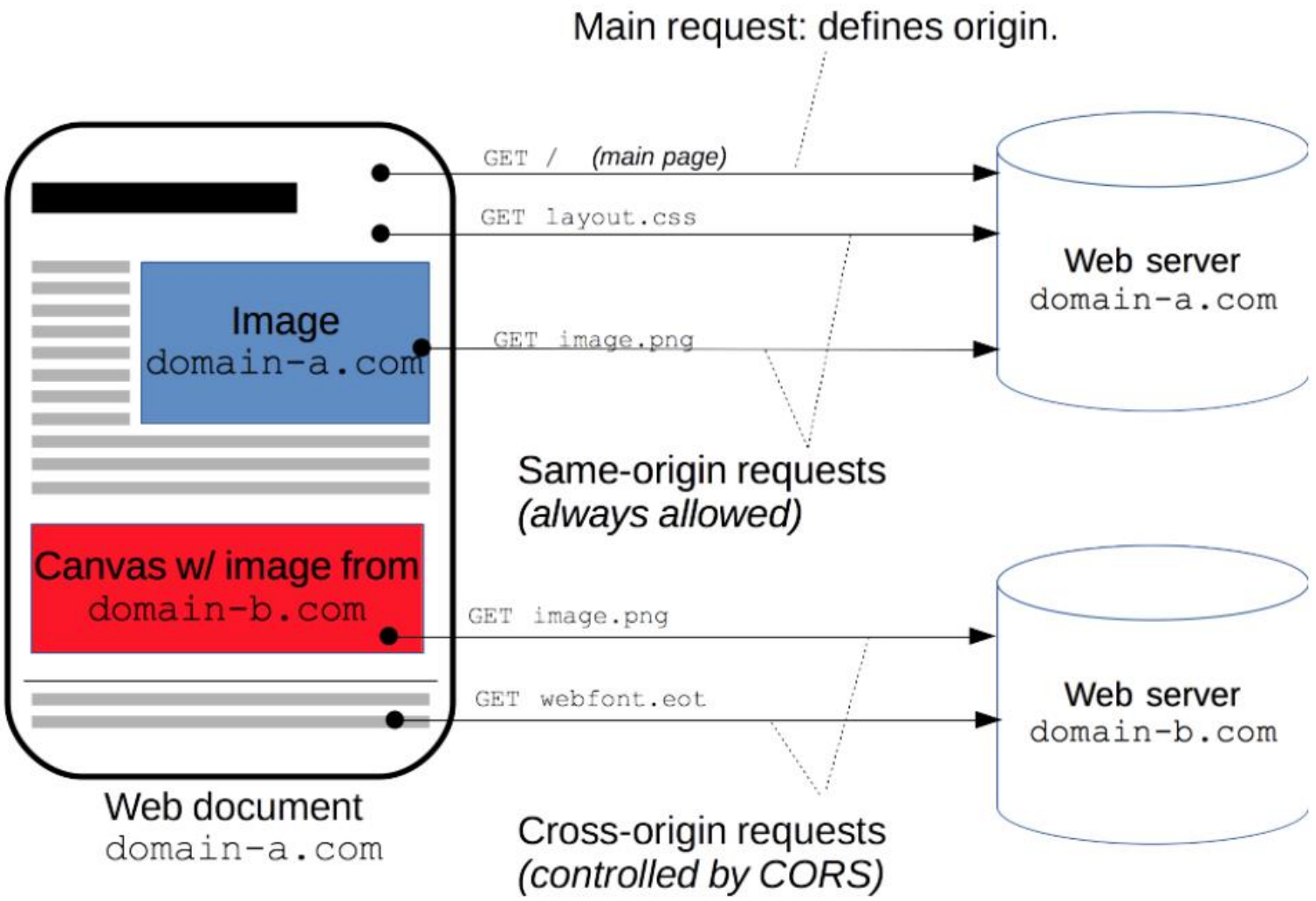
- Envio do pedido
  - Open – Abre a ligação indicada no url
  - Send – Envia o pedido sem parametros

```
xmlHttp = new XMLHttpRequest();
xmlHttp.onreadystatechange=funcTrataResposta;
xmlHttp.open("GET",url);  // assincrono
xmlHttp.send();
```

- Receção da resposta

```
Function funcTrataResposta() {
  if (xmlHttp.readyState==4) {
      document.getElementById("txtHint").innerHTML
              =  xmlHttp.responseText;
  }}
```

AJAX: Cross-Origin Resource Sharing



AJAX: Vantagens

- Helps to build fast, dynamic websites.
- Ajax allows to perform processing on client computer (in JavaScript) with data taken from the server thereby reducing server load by moving a part of server functionality to client side.
- Ajax can selectively modify a part of a page displayed by the browser, and update it without the need to reload the whole document with all images, menus etc. This bridges the gap between desktop and web applications.
- Saves bandwidth by only transmitting new information
- Creates possibility of entirely new types of user interfaces not possible in traditional model.

AJAX: Desvantagens

- Poor compatibility with very old or obscure browsers, and many mobile devices.
- Limited Capabilities like multimedia, interaction with web-cams and printers, local data storage and real time graphics.
- Not everyone have JavaScript enabled. If JavaScript is not activated, Ajax can't works. The user must be asked to set JavaScript from within options of the browser, with the "noscript" tag.
- Too much code makes the browser slow.
- The back button problem. People think that when they press back button, they will return to the last change they made. but in AJAX this doesn not hold.
- Possible network latency problems. People should be given feedback about the processing.
- Does not run on all browsers.